Vol. 1, No. 4 $3.50 in U.S.A.

with pages from LoCAItion THE INTERNATIONAL JOURNAL OF COMPUTER ASSISTED INSTRUCTION

For People Interested in the Enrichment of Personal Computing

# 99er magazine

Covering the TI-99/4A and other 16-Bit Texas Instruments Personal Computer Systems.

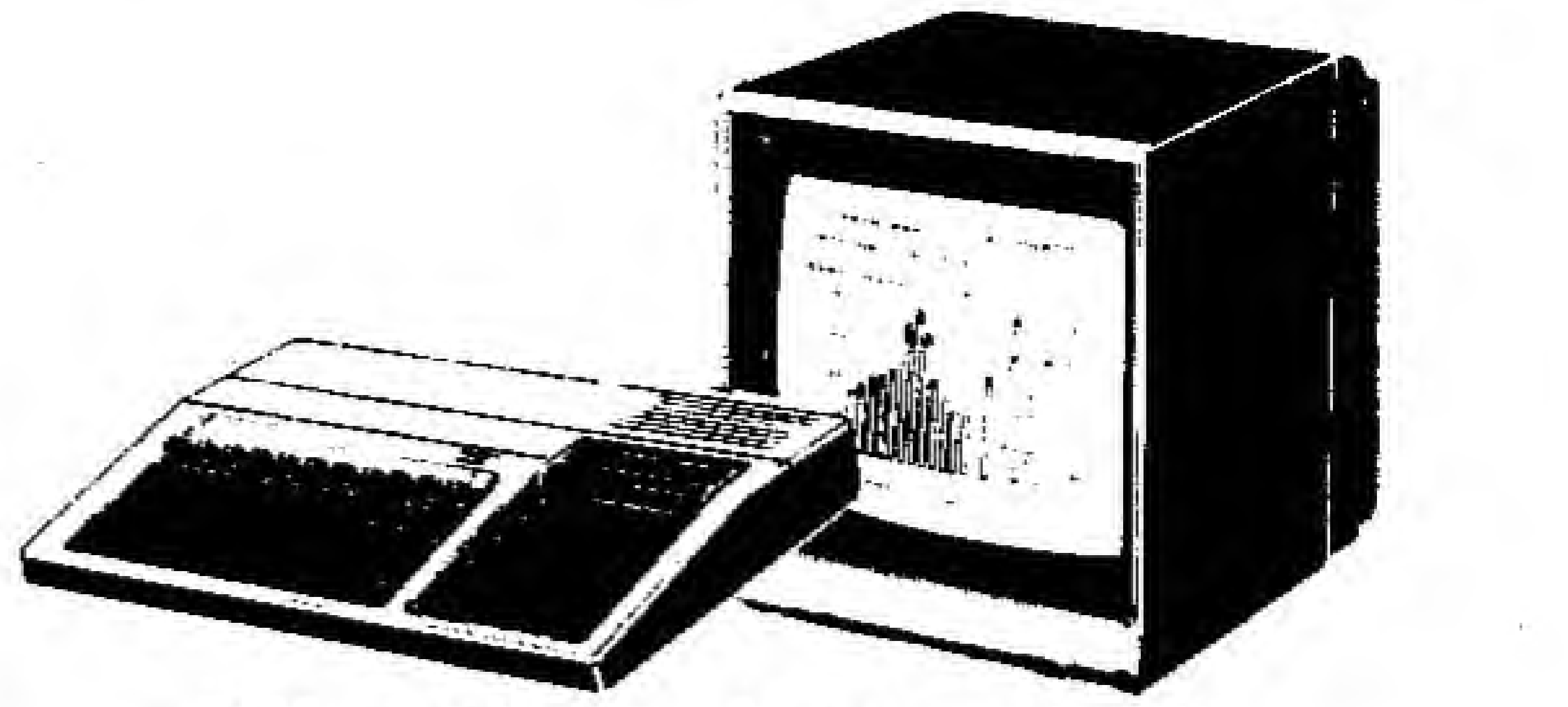
Also Featuring: Logo Times

0882 501501 97381STE100
DONALD STEFFEN
10082 SILVERTON RD
SILVERTON OR 97381

## Computer Gaming:
Breaking Through the Plane of Reality Into the Realm of Imagination

DON STEFFEN

# SAVE BIG ON THE TI-99/4A



**Great savings on all TI-99/4A Products and Accessories**

***"We will meet any price"***
***In this issue, if our competitor has goods on hand.***

## NEW PRODUCTS!

**Peripherals and Accessories**

| | |
|---|---|
| Peripheral Expansion Box | $199.95 |
| ~~RS-232 & Parallel Card~~ | 139.95 |
| Disk Controller Card | 199.95 |
| Expansion Box Disk Drive | 299.95 |
| ~~Memory Expansion Card~~ | 229.95 |
| P-Code Peripheral | 299.95 |
| VCR Controller | 539.95 |
| Panasonic VCR Cable | 89.95 |
| Sony VCR Cable | 89.95 |
| Pioneer VCR Cable | 89.95 |
| Computer Competency Books | ea. 9.95 |

**Command Modules**

| | |
|---|---|
| Personal Report Generator | 44.95 |
| Reading Fun | 49.95 |
| Scholastic Spelling | ea. 49.95 |
| Typing Tutor | 34.95 |
| Adventure (with Diskette) | 44.95 |
| Adventure (with Cassette) | 44.95 |
| Tombstone City: 21st Century | 34.95 |
| TI Invaders | 34.95 |
| Car Wars | 34.95 |
| ~~Munch Man~~ | 34.95 |
| Editor Assembler | 79.95 |
| ~~Mini-Memory~~ | 79.95 |

**Diskettes**

| | |
|---|---|
| Bus. Aides - Invoice Mgmt | 59.95 |
| Bus. Aids - Cash Mgmt | 35.95 |
| Bus. Aides - Lease/Purchase | 59.95 |
| Speak & Spell | 26.95 |
| Speak & Math | 26.95 |
| Bridge Bidding II | 26.95 |
| Bridge Bidding III | 26.95 |
| Spell Writer | 26.95 |
| Teach Yourself Beginning BASIC | 26.95 |
| Saturday Night Bingo | 26.95 |
| Draw Poker | 22.95 |
| Pirate Adventure | 26.95 |
| Adventureland | 26.95 |
| Mission Impossible | 26.95 |
| Voodoo Castle | 26.95 |
| The Count | 26.95 |
| Strange Odyssey | 26.95 |
| Mystery Fun House | 26.95 |
| Pyramid of Doom | 26.95 |
| Ghost Town | 26.95 |
| Savage Island I & II | 36.95 |
| Golden Voyage | 26.95 |

Many of the above games also available on cassette.

## CONSOLE & PERIPHERALS

| | |
|---|---|
| X TI-99/4A Home Computer | 367.95 |
| X Solid State Speech Synthesizer | 109.95 |
| Telephone Coupler (Modem) | 164.95 |
| RS-232 Accessories Interface | 164.95 |
| X Disk Drive Controller (One Disk Manager packed with Disk Controller) | 219.95 |
| X Disk Memory Drive | 367.95 |
| Solid State Printer | 299.95 |
| Memory Expansion (32K RAM) | 289.95 |
| X R. F. Modulator (TV Adapter) | 39.95 |
| 10" Color Monitor | 324.95 |

## OPTIONAL ACCESSORIES

| | |
|---|---|
| X Wired Remote Controllers (Pair) | 26.95 |
| X Dual Cassette Cable | 12.95 |
| Monitor Cable | 17.95 |
| Audio Adapter (Headphone Jack) | 17.95 |
| Thermal Paper (2 pack) | 9.95 |
| Blank Overlays (4 pack) | 7.95 |

## DOCUMENTATION

| | |
|---|---|
| X Beginning BASIC Manual | 9.95 |
| X User's Reference Guide | 9.95 |

## APPLICATION PROGRAMS

**Home Management/Personal Finance**

**Command Modules**

| | |
|---|---|
| X Home Financial Decisions | 24.95 |
| X Household Budget Management | 33.95 |
| Securities Analysis | 45.95 |
| Personal Record Keeping | 39.95 |
| Tax/Investment Record Keeping | 58.95 |
| Personal Real Estate | 58.95 |

**Diskette**

| | |
|---|---|
| Mailing List | 54.95 |
| Personal Financial Aids | 16.95 |
| Checkbook Manager | 16.95 |
| Bus. Aids - Finance Mgmt | 36.95 |
| Bus. Aids - Inventory Mgmt | 58.95 |

**Cassette**

| | |
|---|---|
| Personal Financial Aids | 14.95 |

**Education/Personal Enrichment**

**Command Modules**

| | |
|---|---|
| Early Learning Fun | 24.95 |
| X Beginning Grammar | 24.95 |
| X Number Magic | 16.95 |
| X Video Graphs | 16.95 |
| Video Chess | 54.95 |
| X Physical Fitness | 24.95 |
| Early Reading | 45.95 |
| Music Maker | 36.95 |
| Weight Control and Nutrition | 54.95 |
| Addition and Subtraction I | 34.95 |
| Addition and Subtraction II | 34.95 |
| Multiplication I | 34.95 |
| TI-LOGO | 199.95 |

**Diskette**

| | |
|---|---|
| Teach Yourself BASIC | 28.95 |
| Music Skills Trainer | 25.95 |
| Computer Music Box | 16.95 |
| Market Simulation | 16.95 |
| Teach Yourself Extended BASIC | 20.95 |
| Music Maker Demonstration | 14.95 |
| Basketball Statistics | 22.95 |
| Bridge Bidding I | 24.95 |

**Cassette**

| | |
|---|---|
| Teach Yourself BASIC | 24.95 |
| Music Skills Trainer | 22.95 |
| Computer Music Box | 14.95 |
| Market Simulation | 14.95 |
| Teach Yourself Extended BASIC | 16.95 |
| Bridge Bidding I | 20.95 |

**Entertainment**

**Command Modules**

| | |
|---|---|
| X Football | 24.95 |
| Video Games I | 24.95 |
| Hunt the Wumpus | 20.95 |
| Indoor Soccer | 24.95 |
| Mind Challengers | 22.95 |
| A-Maze-Ing | 20.95 |
| The Attack † | 33.95 |
| Blasto † | 20.95 |
| Blackjack and Poker | 20.95 |
| Hustle † | 20.95 |
| Zero Zap † | 16.95 |
| Hangman | 16.95 |
| Connect Four † | 16.95 |
| Yahtzee † | 20.95 |

**Diskette**

| | |
|---|---|
| X TI-Trek (w/speech) | 12.95 |
| Mystery Melody | 14.95 |
| Oldies But Goodies - Game I | 18.95 |
| Oldies But Goodies - Game II | 22.95 |

## OTHER APPLICATION PROGRAMS

**Command Modules**

| | |
|---|---|
| X Diagnostic | 27.95 |
| X Demonstration | 59.95 |
| X Speech Editor | 36.95 |
| Statistics | 37.95 |
| X Extended BASIC | 76.95 |
| Terminal Emulator II | 39.95 |

**Diskette**

| | |
|---|---|
| Programming Aids I | 12.95 |
| Programming Aids II | 20.95 |
| Math Routine Library | 24.95 |
| Electrical Engineering Library | 24.95 |
| Programming Aids III | 16.95 |
| Graphing Package | 16.95 |
| Structural Engineering Library | 24.95 |

**Cassette**

| | |
|---|---|
| Programming Aids I | 9.95 |
| Math Routine Library | 20.95 |
| Electrical Engineering Library | 20.95 |
| Graphing Package | 14.95 |
| Structural Engineering Library | 20.95 |

† Attack, Blasto, Hustle, Zero Zap, Connect Four and Yahtzee are trademarks of Milton Bradley.

32 K Expansion requires use of Extended BASIC or TI-LOGO

Speech Synthesizer requires use of Speech Editor or Terminal Emulator II

**tam's**
INCORPORATED
*14932 GARFIELD AVENUE*
*PARAMOUNT, CA 90723*

VISA MasterCard

**Order Toll-Free**
**800-421-5188**

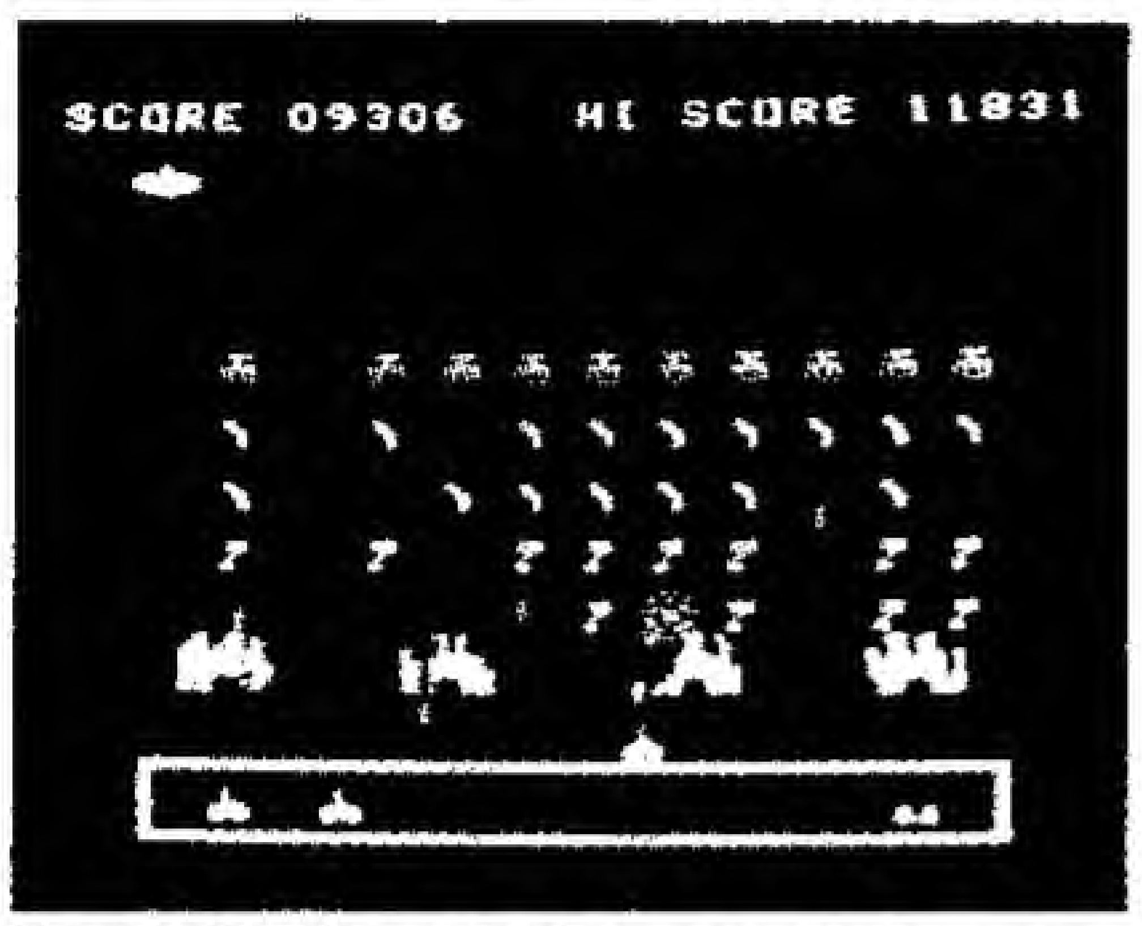
**In California call (213) 633-3262**

**Payment by check:**
For faster delivery use cashier's check or money order (personal checks take 3 weeks to clear). Add shipping charges: 1% of your order ($3.75 minimum). East of Mississippi River add $1.50. California residents add 6% sales tax. Subject to availability. USA prices only.

**Information line (213) 633-3262**

Fifty Years of Innovation

Texas Instruments Personal Computer



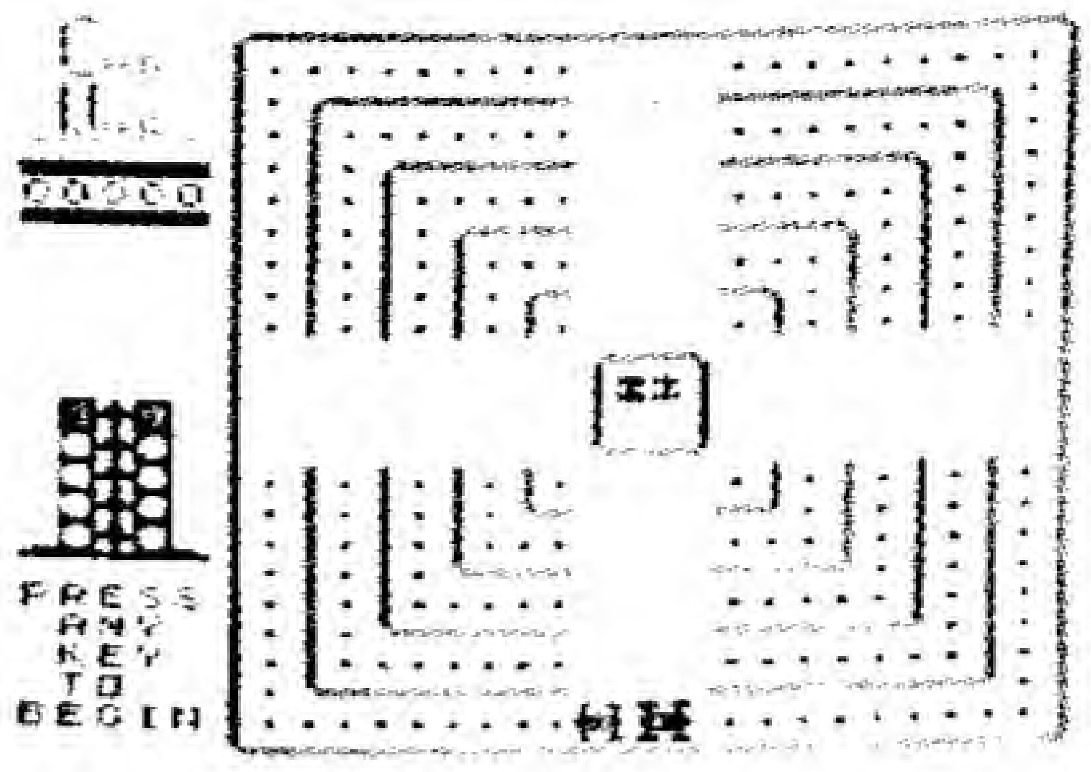
Now with the exciting new TI Invaders game, you can try to destroy nasty space creatures before they demolish your missiles!

## TI Invaders

What if your world was under attack by numerous downright nasty creatures from space? This is the situation you face with the exciting new TI Invaders Command Module. It's up to you—can you save your world from this hostile horde? Can you destroy these swarming invaders before they demolish your missile?

You must use your wit and quick movements in this one-player game to destroy the multi-colored creatures. But, as you shoot down each attacker, the swarming horde increases its speed and sprays shots at your missile. Your missiles are limited, so be cautious!

The TI Invaders Command Module (model number PHM 3053) has a suggested retail price of $39.95. The TI Wired Remote Controllers are optional.



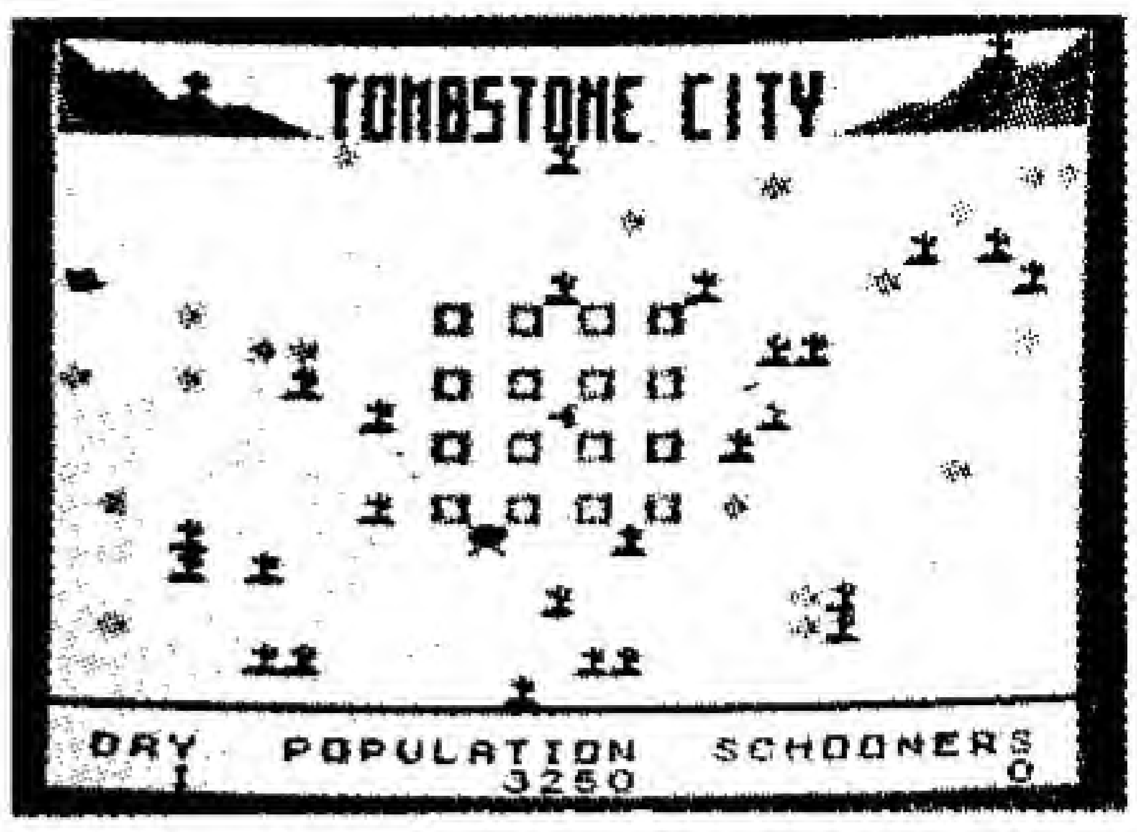
Car Wars, one of TI's many new arcade games, pits your driving expertise and agility against that of the computer.

## Car Wars

If you enjoy the excitement of high-speed racing, combined with the challenge of out-maneuvering a canny opponent, then Car Wars is your kind of game. With the Car Wars Command Module, you pit your speed and skill against the computer's as you try to get your car around the track without getting "crashed" off the field.

With the Car Wars module, you try to score points by clearing as many dots as possible from the lanes. In addition, you also want to avoid the computer's yellow car and obtain extra cars by clearing all of the driving lanes on the display. Various levels of difficulty allow you to add even more excitement to the action of this one-player game.

The Car Wars Command Module (model number PHM 3054) is one of the many arcade games now available for your TI Home Computer. Its suggested retail price is $39.95. The TI Wired Remote Controllers are optional.



TI's fascinating new Tombstone City: 21st Century game challenges your strategy and skill in stopping a hoard of green alien monsters.

## Tombstone City: 21st Century

Your instinct for survival is challenged immediately with the Tombstone City: 21st Century *Solid State Software*™ Command Module. You find yourself in a 21st Century Old West ghost town threatened by an invading hoard of green alien monsters. These villainous creatures called morgs live off of only two things—tumbleweeds and people.

It's up to you and your security force of prairie schooners to stop the morgs before they infest the earth. In accomplishing this goal, you can score points by destroying alien morgs and wiping out tumbleweeds. Be sure and watch out for adjacent pairs of saguaro cactuses—it's from here that a new morg is generated!

A one-player game, Tombstone City is designed to test your strategy as well as your skill. The Command Module version (model number PHM 3052) has a suggested retail price of $39.95. The TI Wired Remote Controllers are optional.

MORE AND BETTER SOFTWARE
MAKES YOUR INVESTMENT A VALUE THAT GROWS AND GROWS.
NEW YEAR SPECIAL any of the above 26.95
Master Charge, Visa, money order or any good payment.
Add 2.98 shipping and handling any size order continental USA.

We carry the complete Texas Instruments personal computer line, hardware and software. Write for our very competitive price list including TEXNET℠

℠ TEXNET is a servicemark of Texas Instruments Incorporated.

**A ACE Computer**

7 W. Airline Hwy.
Corner of Hwys. 57 & 63
Waterloo, Iowa 50701
Phone: 319/236-3861

# 99'er magazine™

**THIS ISSUE'S COVER**
Don Fexer's cover painting depicts how—with a little help from our computer friends—we can break through the plane of every-day reality into the imaginative world of fantasy, action, and high adventure. The video on screen signifies the recursive role of each participant—as both director and stage player under self-direction.

**Publisher/Editor**
Gary M. Kaplan

**Technical Editors**
William K. Balthrop
David G. Brader
John Clulow
Mike Kovacich
G. R. Michaels
Patricia Swift

**Program Editor**
Cheryl Whitelaw

**Office Manager**
Pat Kaplan

**Circulation & Fulfillment**
Rod Hoyle
Benjamin Kaplan
Coleen Peplow

**Advertising**
Patana Ratanapreux
*Tel. 503-485-8796*

**Contributing Editors**
F. T. Berkey
Norma Clulow
Henry Gorman, Jr.
Roger Kirchner
Mark Moseley
James Muller
Samuel Pincus
Regena
George Struble
Dennis Thurlow
Jerry Wolfe

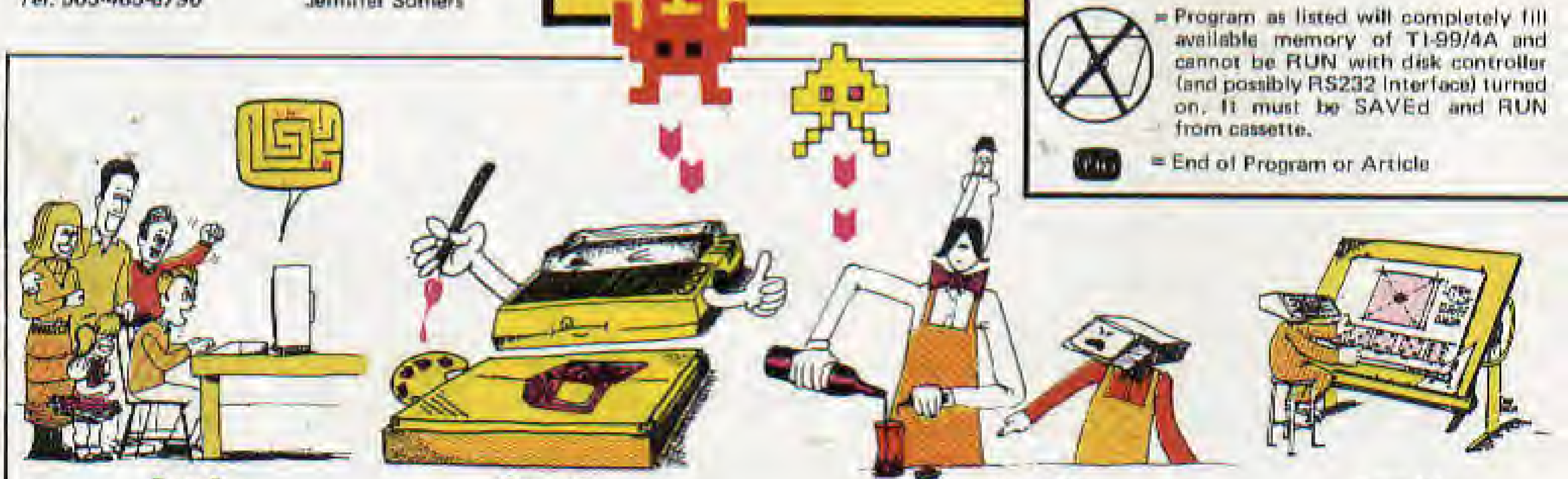
**Production Manager**
Norman Winney, Jr.

**Production & Design**
Greg Davis
Corby Poticha
Carl Shaw
Jennifer Somers

**Programming Conventions**

⊗ = Program as listed will completely fill available memory of TI-99/4A and cannot be RUN with disk controller (and possibly RS232 Interface) turned on. It must be SAVEd and RUN from cassette.

■ = End of Program or Article



Page 9 Page 17 Page 39 Page 24

**A Resource for People Interested in the Enrichment of Personal Computing**

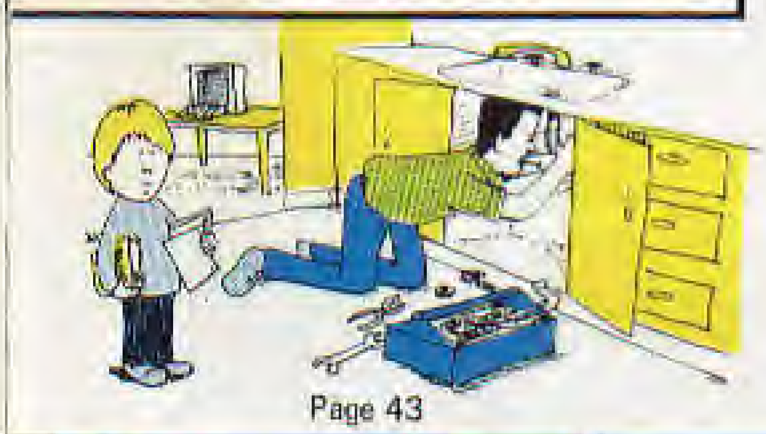
Volume 1 No. 4

THE MAGAZINE OF THE LOGO LANGUAGE — LOGO TIMES™

```
99'ER VERSION 1 . 4 . 1 X B M
volume no. ───────┘   │   │ │ │
issue no. ────────────┘   │ │ │
version ──────────────────┘ │ │
   1 = original program     │ │
   2 }                      │ │
   : } = no. of update      │ │
   n }                      │ │
TI Extended BASIC ──────────┘ │
Expansion Memory Required ────┘
```

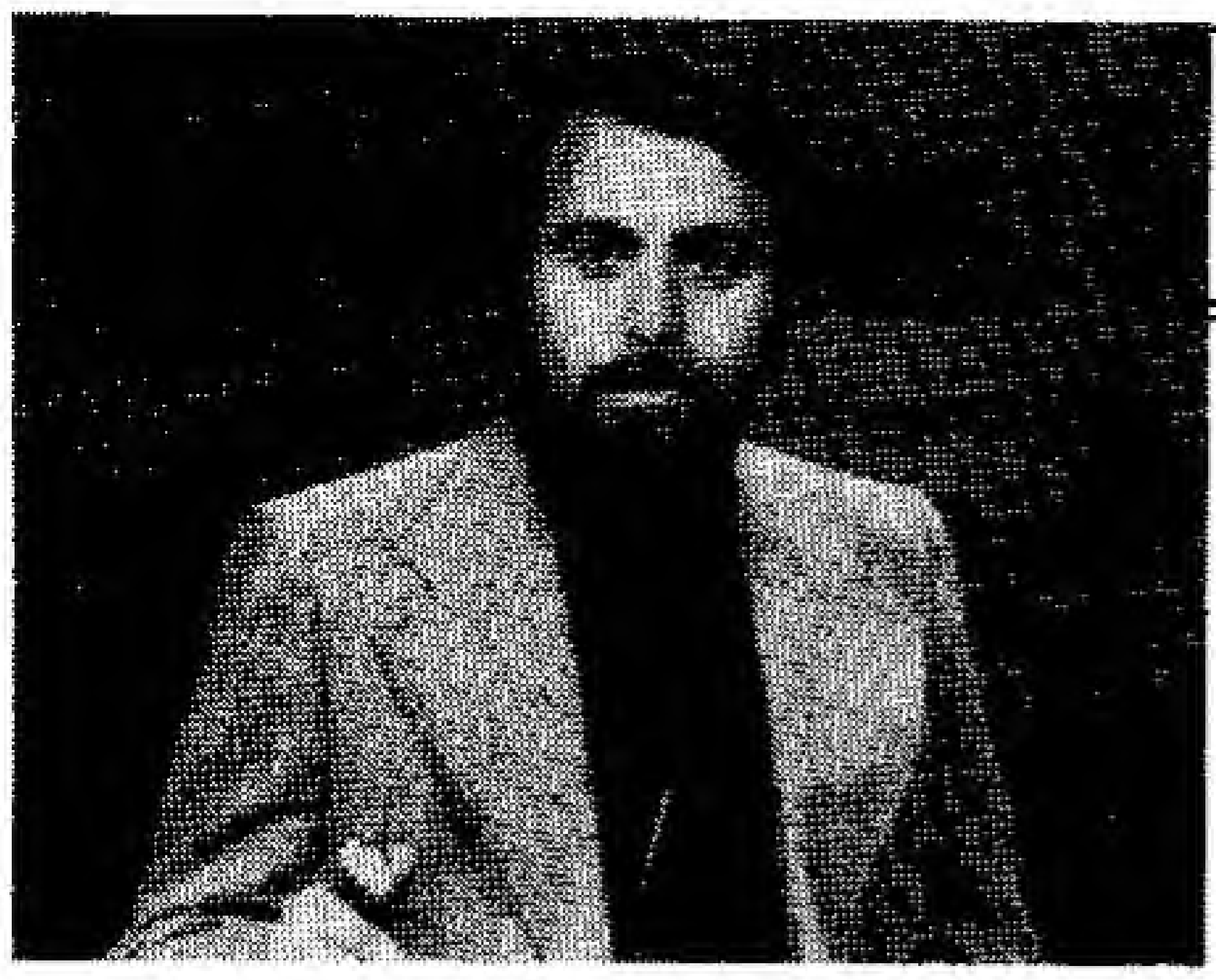
**99'er Magazine** is published bimonthly by Emerald Valley Publishing Co., P.O. Box 5537, Eugene, OR 97405. The editorial office is located at 2715 Terrace View Drive, Eugene, OR 97405 (Tel. 503-485-8796). Subscription rates in U.S. and its possessions are $18 for one year, $32 for two years, and $45 for three years. In Canada and Mexico $22 for one year, $42 for two years, $60 for three years. Other foreign countries $28 for one year surface, $43 for one year air delivery. Single copy price in U.S. and its possessions is $3.50, and $4.00 in Canada and Mexico. Foreign subscription payment should be in United States funds drawn on a U.S. bank. Application to mail at controlled circulation postage rates is pending at Eugene, OR 97401. POSTMASTER: Send address changes to **99'er Magazine**, P.O. Box 5537, Eugene, OR 97405. Subscribers should send all correspondence about subscriptions to above address.

Address all editorial correspondence to the Editor at **99'er Magazine**, 2715 Terrace View Drive, Eugene, OR 97405. Unacceptable manuscripts will be returned if accompanied by sufficient first class postage and self-addressed envelope. Not responsible for lost manuscripts, photos, or program media. Opinions expressed by the authors are not necessarily those of **99'er Magazine**. All mail directed to the "Letters to the Editor" column will be treated as unconditionally assigned for publication, copyright purposes, and use in any other publication or brochure, and are subject to **99'er Magazine's** unrestricted right to edit and comment. **99'er Magazine** assumes no liability for errors in articles or advertisements. Mention of products by trade name in editorial material or advertisements contained herein in no way constitutes endorsement of the product or products by **99'er Magazine** or the publisher unless explicitly stated.


Page 43

Page 33

Page 34



# ON SCREEN

**By Gary M. Kaplan**
**Editor & Publisher**

To paraphrase an old cliché, "All work and no play makes a computer a dull machine." By this yardstick, most computers were once (not too long ago) *very* dull machines. With the advent and proliferation of *personal* computers used in the home, however, the balance has changed: Game playing is now an important part of many a computer's repertoire, and for an increasing number of consumers, is becoming a valid enough reason in itself to buy a computer. Find out what this new games phenomenon is all about in our leadoff article, *The Joy of Computer Gaming.*

If after reading that article, you're game for a little fun and excitement, key in (and save on tape) any of the program listings for the three *Games of Time & Direction* found in *Kelley's Korner.* Whether you race through mazes, rescue trapped lunar miners, or dodge cars on the freeway, you're sure to have a great time. And if you're a games programmer, you may even pick up a few tips in the process.

Speaking about programming tips, if some of you beginning programmers are interested in learning how to design and write your own computer game, be sure to carefully read our *Starting from Square One* section. And for the advanced programmers, we have the second part in our tutorial series on *How to Write a BASIC Program that Writes BASIC Programs.* But if programming games in Assembly Language is more your "speed," check out *Super Language, Part 2*; it should prove interesting fare and whet your appetite for the new Editor/Assembler package from TI.

When guests drop in unexpectedly to play all your computer games, there's sure to be a time-out when some refreshments would be welcome. When this happens, we've got just the solution for you: Load up our *Micro Bartender* program and let your computer do the hard part . . .

If during the afternoon or evening of entertainment, your guests can't stop asking questions about your remarkable home computer, don't be too surprised. But don't worry. After reading our center-spread photo feature on *TI at the Winter Consumer Electronics Show,* you'll be able to field most of their questions.

No matter how much fun it will all be, however, there'll come a time to put away the games, say goodbye to guests, and once again start *Getting Down to Business* with, for example, our introductory article on computer inventory. Following it up with *From Dots to Plots* will ensure that you're well versed in bit-plot printer graphics — knowledge that should prove useful for custom printing and plotting applications in science, engineering, business, or even for designing T-shirt art! And by all means, make sure you learn *The Secret of Personal Record Keeping* before you get too involved with our many other features and sections.

For the education-minded, we have a couple of surprises: First, there's our new look to *Pages From OnLoCAItion.* And in this issue, we will take you on location to a music class in Ohio where a Texas Instruments microcomputer is working wonders. You'll be able to see just for yourself how easily it's done when you RUN the accompanying program. We've also launched our coverage of microcomputing for the handicapped with a thought-provoking article and colorful example program. Follow that up with another in our series of *Homework Helpers* for checking a student's math assignments, and I think you'll readily agree that some of the best CAI software is found right within the pages of this magazine.

The big education-oriented change in this issue, however, is the new "magazine-within-a magazine." *LOGO Times* is the first of its kind—a magazine devoted to the magic and adventure of a creative computer learning environment rather than the drill-and-practice type of CAI software found in OnLoCAItion. Learn what TI LOGO is all about in what we believe to be an extremely important supplement to *99'er Magazine.*

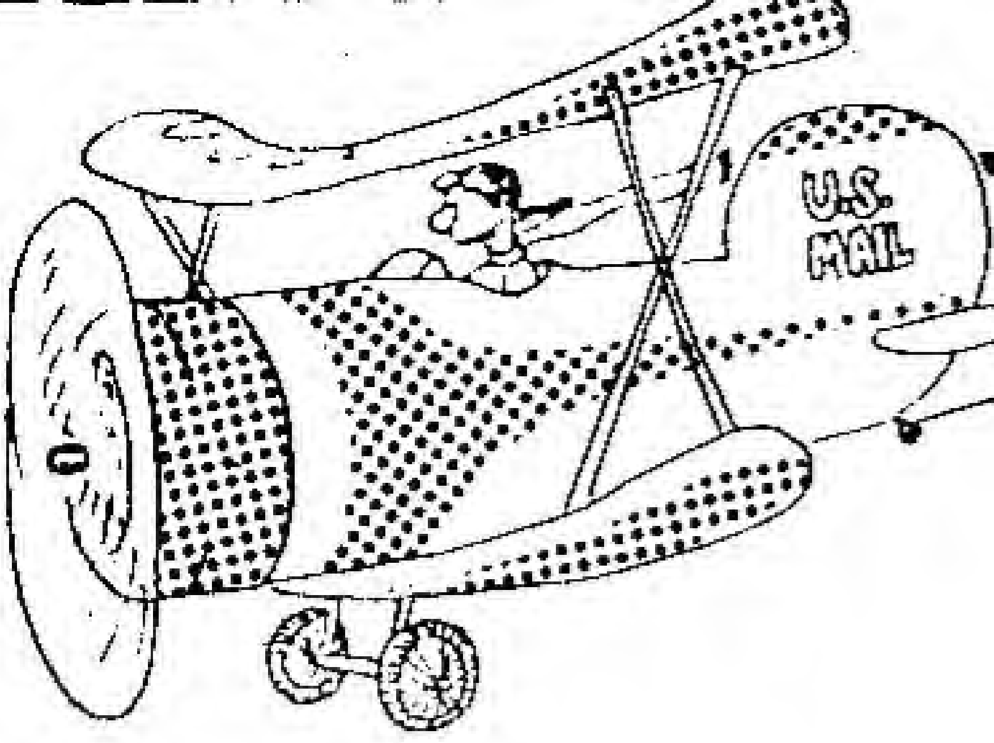
This issue also marks the debut of another new feature, the *99'er Road Map* (page 8). We've designed this as a multiple guide for (1) Texas Instruments TI-99/4 and TI-99/4A users who are beginners to microcomputing, (2) more experienced users and programmers who are new to our magazine, and (3) newcomers to personal computing who don't as yet have their own microcomputer. We're hoping that the suggested pathways, glossary, and references to related articles in past issues will contribute to a better understanding of article content, and allow more of you to make faster progress along your individual journeys of discovery and enrichment. So please let us know how you like this new guide, and in what ways we can improve it.

As many of you have undoubtedly noticed by now, there are no longer any dates of issue appearing on the cover or contents pages. This issue is simply referred to as Volume 1, Number 4. The next issue you receive (approximately two months after you've received this one, so jot down the dates) will be Volume 1, Number 5, and so on. After Number 6, our first year of publication will come to a close, and Volume 2, Number 1 will be on its way. As soon as we can garner enough subscribers (and we need your HELP on that one), you can expect a monthly publication.

On another front, I'm happy to announce that we've acceded to your numerous requests to make our magazine software available on tapes. (See page 93 for details.) This decision was prompted in part by our expectation that some subscribers might have a more difficult time reading and keying in the reduced -size program listings that you'll find in this issue. Incidentally, by reducing the amount of space these programs take up —*without* cutting down on the number of programs—we were able to "shoe -horn" an additional ten pages of article material into the magazine. Please let us know how you like our new "compressed" look of program listings.

My final announcement is about *99'er TI-Fest.* This is a trade show, conference, and seminar for everyone interested in Texas Instruments personal computers and learning aids. We're excited to be presenting the event, and I hope that when you read all the details on page 92, you'll be excited also. Watch forthcoming issues for additional information about *The* event of the year . . . one you won't want to miss. So pass along the word—*99'er TI-Fest is coming!*

**Until next issue—Have fun reading, learning, and RUNing.**



# LETTERS TO THE EDITOR

Dear Sir:
Just thought I'd pass this along to you and your readers. I was in the process of typing a long program into my 99/4A when I received a MEMORY FULL error after ENTERing one line near the end of the program. For backup purposes, I saved this work on cassette before doing any editing. Later, I tried to copy this program back off of the cassette. As soon as the machine finished reading, and tried to write "DATA OK", my video display crashed (in my case, my TV). Recovery was impossible without turning the machine off.
Playing a hunch, I copied the program off the cassette again. With the display down, I managed to delete the last few lines of the program, and copied it back to cassette. This shortened form could then be read without crashing the display. A word to the wise, as they say.
By the way, congratulations on a fine magazine. I've seen some of the other "computer" magazines, and wouldn't waste my money on them.

Neal Nesselbeck
Kenmore, NY

Dear Sir:
I would like to hear more about the personalities in the TI world. How about some of the ideas and thoughts of key people in TI. What about sponsoring a round table of sort where key people sit down and air their thoughts, gripes, hopes about where the TI world is now, is going, or should go. The right people, perhaps in a hotel room at some show should generate an article that would make national news. I would like to hear more than I can find about the TI Business System recently announced . . . the 99/4 software should be able to run on the new system and vice versa.

Jim Horn
Falls Church, VA

*Your letter is very timely, Jim. See our announcement of* **99'er TI-Fest** *in this issue. As for the potential software compatibility, the UCSD Pascal System is the magic link that brings software portability to hardware of three separate TI Groups. Watch for more on this in future issues.*

Dear Sir:
I have an idea about REGENA: Because the amount of information which comes from REGENA is varied and worldwide I feel that REGENA is not one person but many and that the name REGENA is an acronym of perhaps the initials of those involved. Just a thought.

R. Learned
Raymond, NE

*At one time, we suspected the same. Only, our latest startling package squelched all possibility of that. Hear the complete, incredible story about it in the next issue. Meanwhile, check out the "Tips . . . " column for some more of REGENA's wisdom.*

**PLEASE DON'T FORGET TO RETURN THE QUESTIONNAIRE ON THE FRONT BIND-IN CARD.**

Dear Sir:
Could you suggest a book that explains the "OPEN" file process. The two basic books *Programming BASIC* by Peckham and *Beginners Basic* [that comes with the computer] are extremely inadequate in this area. And is there a good book of example graphics? I don't draw well and it seems as if this would be a good subject for someone.
Lastly, keep up the good work !! I purchased my computer without knowing about you, but after seeing many other "computer" magazines, you are light years ahead of the pack. Thanks.

Tim Williams
Gig Harbor, WA

*You might want to check out Introduction to TI BASIC (available from the 99'er Bookstore). Also, watch forthcoming issues for a series of tutorial articles that cover the topics you mentioned.*

Dear Sir:
Would it be possible to get tapes of the programs you print in the *99'er Magazine*? It takes so long to try to type them into the computer for use and storage.
You could put together packages of programs to be sold to subscribers at a modest cost. The value of a Home Computer is in its use. It would be a great service and a lot of fun for your readers to more fully utilize their 99/4 computers.
The more 99/4 owners that use their computer, the more enthusiastic they become about telling their friends about the Home Computer and your 99'er Magazine.
Thanks for a great magazine.

Ronald B. LaVergne
Deerfield Beach, FL

*O.K., Ron. You and hundreds of other readers who have written to us with this same request finally win! Look for the details near the back of this magazine.* (See Advertisement on p. 93)

Dear Sir:
First, I would like to congratulate you on the production of your excellent magazine. I had considered *BYTE* the most useful of the genre' until I saw yours. Absolutely the most valuable source of ideas (and suppliers) for those of us using a TI-99 Home System.
Keep up the good work at the magazine, and if I don't find a renewal in my Christmas stocking, I'll send one in myself.

Dennis E. Clouse
Concord, CA

*Thanks for your kind words, Dennis. You and our other loyal readers can help us out of a sticky problem. It's one of visibility: There are many thousands of new TI-99/4A owners out there who just don't know that we exist! So how about mustering the troops and getting the word out. To continue to bring you the quality magazine you deserve, we must rapidly increase both our circulation and advertising support.*

Dear Sir:
Thanks for the very informative article *Frugal Floppies.* Another piece of non-TI hardware that I use is the Radio Shack Direct Connect Modem. The modem was $145 and I made my own cable with plugs and cable purchased at Radio Shack. Only five wires are used and the correct connections for the plugs can be found in the instruction manual packed with the TI RS232 interface. The modem works quite well.

Bob Medley
Minot AFB, ND

Dear Sir:
I'm very pleased with your magazine —finally TI RULES! Please send me your author's guidelines for software that I wrote. Please hurry with a reply and my next issue, My first issue is all read out (10 times over!)

Paul Tyma
Broadview Hts., OH

*Yours is the type of letter, Paul, that we like to show to advertisers when they ask us about "average time our readers spend with each issue."*

Dear Sir:
I just wanted to let you know how much I enjoyed issue # 3. Not being a programmer I found the section on conversion between BASICs quite helpful. After reading it I was able to rewrite a program I had been working on for several months.
My TI-99/4 is so easy to use that even my 4 year old son is using it. The Computer Assisted Instruction programs [found in the OnLoCAItion pages] are truely marvelous.
Keep up the great work. I am anxiously awaiting my next issue.

Bill Heeper
Huber Hts., OH

Dear Sir:
In the September/October issue of *99'er Magazine*, REGENA (page 85) gave some information on randomizing. Having played with the randomizer before, I was not surprised to note that her results were not only similar, but indeed predictable. To substantiate my suspicions, I ran REGENA's same program on my 99/4 several times for 1000000 (that's right . . . 1 million) numbers. For 1,000,000 RND's the numbers 1 to 10 will be randomized to 10.1%, 10.0%, 10.3%, 10.3%, 9.8%, 10.1%, 10.0%, 10.2%, 9.6%, and 9.6% respectively every run.
However, to get true randomness instead of the predictable pseudo random numbers, you simply perform the RANDOMIZE statement each time you use the RND statement . . . instead of only once at the beginning of the program. When I did this (also for several million runs) the results were never the same and were *more fair*, i.e., the highest percent was 10.2% and the lowest was 9.7%.
I was also interested to know whether the RND statement would ever give back the same number (i.e., the same 10 digits behind the decimal) . . . as the first number it happened to RND to. I also coded that logic into the same runs as above and the results were as follows:
— When the RANDOMIZE statement appears only at the beginning of the program and is executed once, the first number RND'd to was never RND'd to again in the same run of 1,000,000 RND's.
— But, when the RANDOMIZE statement is executed each time that the RND statement is, the first number RND'd to is again RND'd to from 1 to 10 times (it varies) in a run of 1,000,000 RND's.
My method does take longer (to randomize each time), but it gives true randomness.
P.S. The 99'er Magazine is great. I look forward to each copy.

L. G. Mayer
Davenport, IA

Continued on p. 75

## HOW TO READ THIS MAGAZINE

This road map is a reader aid. It suggests alternate pathways through the magazine's contents that are appropriate for different groups of readers. Since the articles cover a wide range of interest and are aimed at readers with all levels of experience—from beginners wanting to learn more before purchasing a microcomputer, all the way on up to professional programmers—additional reference information to help bridge the knowledge gap was also placed on the map.

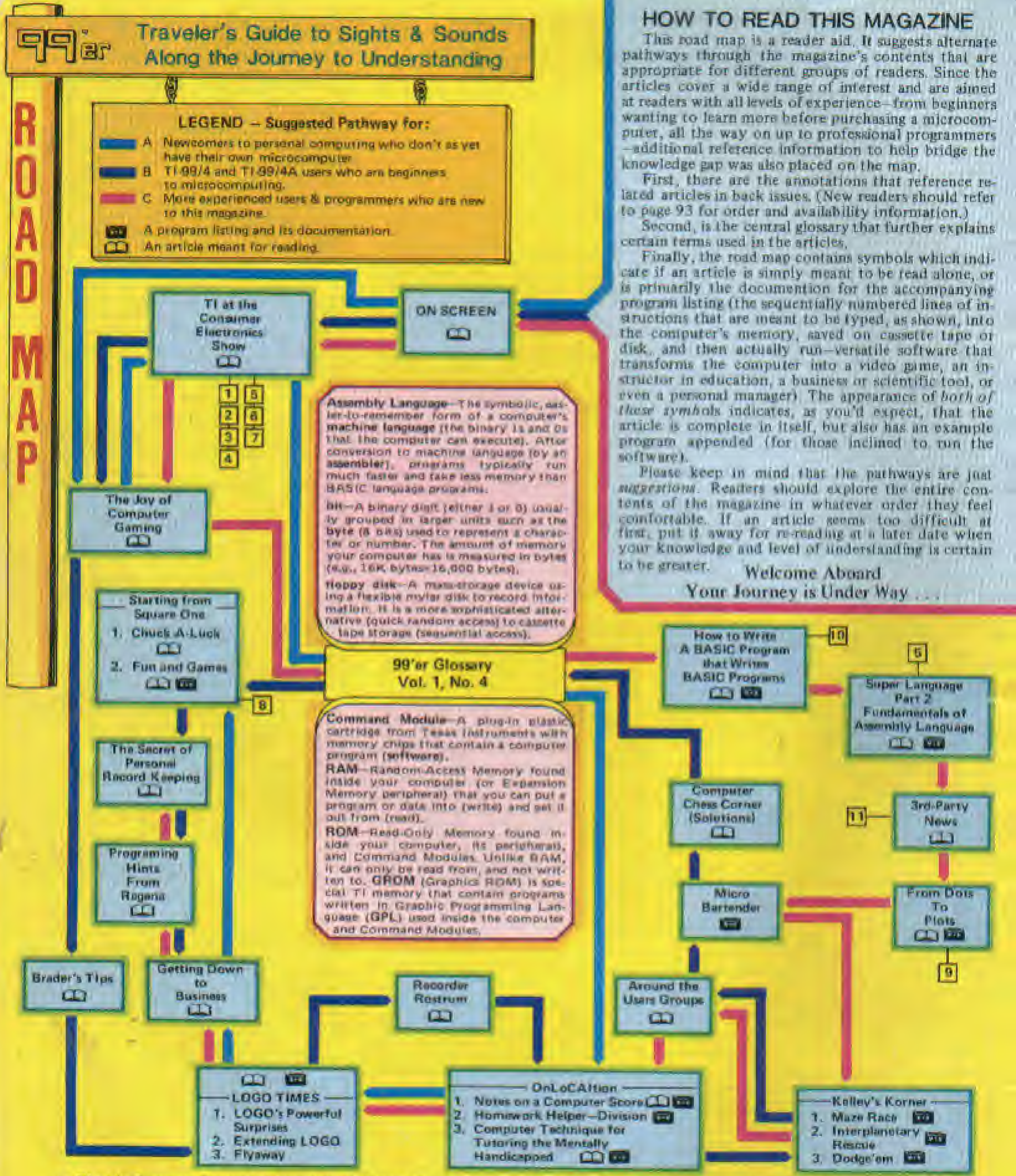
First, there are the annotations that reference related articles in back issues. (New readers should refer to page 93 for order and availability information.)

Second, is the central glossary that further explains certain terms used in the articles.

Finally, the road map contains symbols which indicate if an article is simply meant to be read alone, or is primarily the documention for the accompanying program listing (the sequentially numbered lines of instructions that are meant to be typed, as shown, into the computer's memory, saved on cassette tape or disk, and then actually run—versatile software that transforms the computer into a video game, an instructor in education, a business or scientific tool, or even a personal manager). The appearance of *both of these symbols* indicates, as you'd expect, that the article is complete in itself, but also has an example program appended (for those inclined to run the software).

Please keep in mind that the pathways are just *suggestions*. Readers should explore the entire contents of the magazine in whatever order they feel comfortable. If an article seems too difficult at first, put it away for re-reading at a later date when your knowledge and level of understanding is certain to be greater.

Welcome Aboard
Your Journey is Under Way . . .



99'er
Traveler's Guide to Sights & Sounds Along the Journey to Understanding
ROAD MAP
LEGEND — Suggested Pathway for:
A Newcomers to personal computing who don't as yet have their own microcomputer.
B TI-99/4 and TI-99/4A users who are beginners to microcomputing.
C More experienced users & programmers who are new to this magazine.
A program listing and its documentation.
An article meant for reading.
TI at the Consumer Electronics Show
ON SCREEN
1 2 3 4 5 6 7
The Joy of Computer Gaming
Starting from Square One
1. Chuck-A-Luck
2. Fun and Games
8
The Secret of Personal Record Keeping
Programing Hints From Regena
Brader's Tips
Getting Down to Business
Recorder Rostrum
LOGO TIMES
1. LOGO's Powerful Surprises
2. Extending LOGO
3. Flyaway
OnLoCAItion
1. Notes on a Computer Score
2. Homework Helper—Division
3. Computer Technique for Tutoring the Mentally Handicapped
99'er Glossary Vol. 1, No. 4
How to Write A BASIC Program that Writes BASIC Programs
10
5
Super Language Part 2 Fundamentals of Assembly Language
Computer Chess Corner (Solutions)
11
3rd-Party News
Micro Bartender
From Dots To Plots
9
Around the Users Groups
Kelley's Korner
1. Maze Race
2. Interplanetary Rescue
3. Dodge'em

**Assembly Language**—The symbolic, easier-to-remember form of a computer's **machine language** (the binary 1s and 0s that the computer can execute). After conversion to machine language (by an **assembler**), programs typically run much faster and take less memory than BASIC language programs.

**bit**—A binary digit (either 1 or 0) usually grouped in larger units such as the **byte** (8 bits) used to represent a character or number. The amount of memory your computer has is measured in bytes (e.g., 16K bytes=16,000 bytes).

**floppy disk**—A mass-storage device using a flexible mylar disk to record information. It is a more sophisticated alternative (quick random access) to cassette tape storage (sequential access).

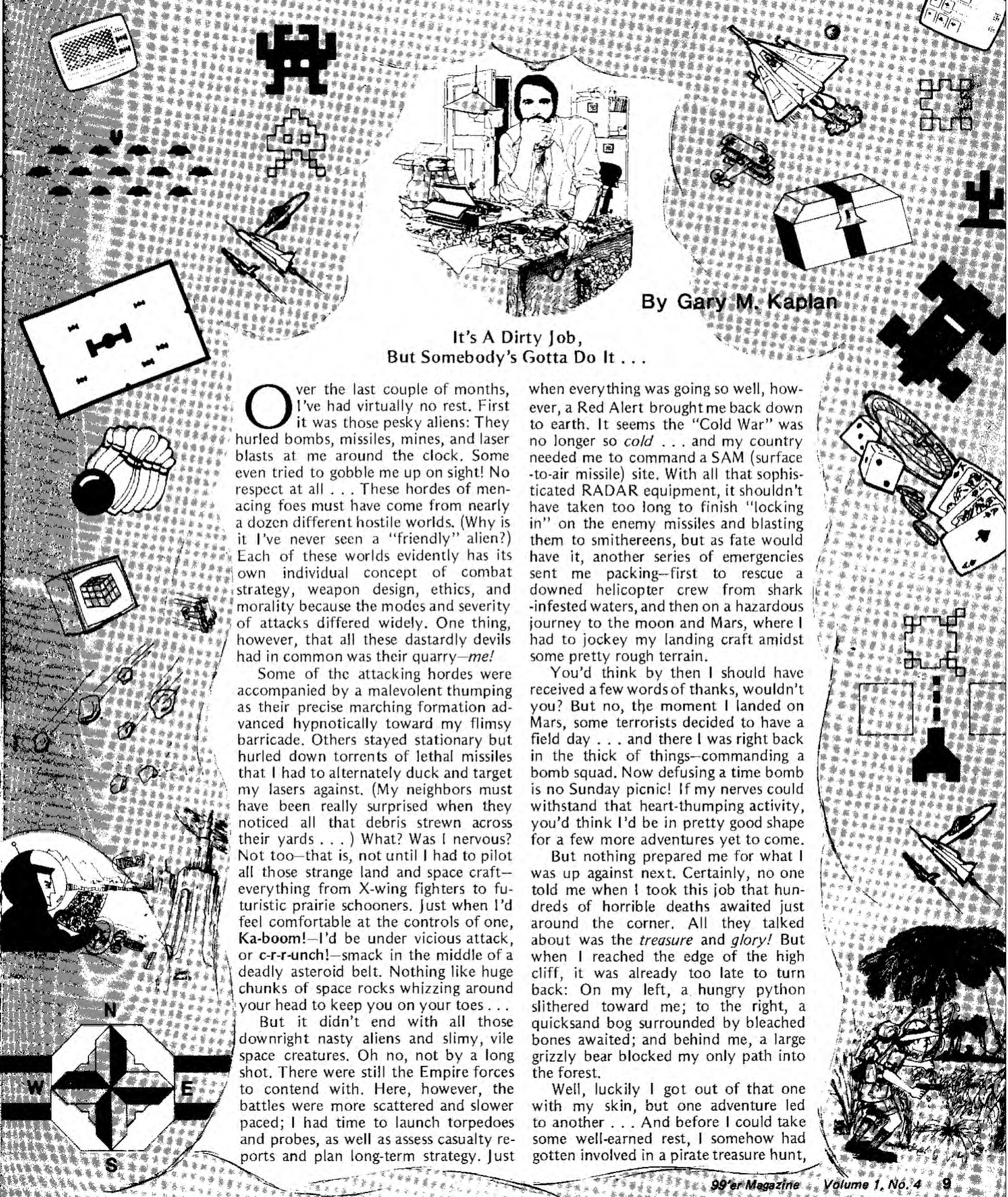
**Command Module**—A plug-in plastic cartridge from Texas Instruments with memory chips that contain a computer program (**software**).

**RAM**—Random-Access Memory found inside your computer (or Expansion Memory peripheral) that you can put a program or data into (write) and get it out from (read).

**ROM**—Read-Only Memory found inside your computer, its peripherals, and Command Modules. Unlike RAM, it can only be read from, and not written to. **GROM** (Graphics ROM) is special TI memory that contain programs written in Graphic Programming Language (**GPL**) used inside the computer and Command Modules.

1 = *TI at the Consumer Electronic Show*; Vol. 1, No. 2; p. 48
2 = *Information Utilities and the Electronic Cottage*; Vol. 1, No. 2; p. 8
3 = *What is UCSD Pascal . . . And Why is Everybody Talking About It?*: Vol. 1, No. 1; p. 52
4 = *The Third-Party Pascal Development System: Some User Impressions*; Vol. 1, No. 3; p. 29
5 = *Super Language: A Preliminary Look at TI's Editor/Assembler*; Vol. 1, No. 3; p. 54
6 = *Computer Assisted Instruction: The State of the Art*; Vol. 1, No. 3; p. 54
7 = *Text-To-Speech on the Home Computer*; Vol. 1, No. 1; p. 30
8 = *How To Write Your Own Programs*; Vol. 1, No. 1; p. 9
9 = *Programming Printer Graphics*; Vol. 1, No. 1; p. 23
10 = *How to Write a BASIC Program that Writes BASIC Programs*; Vol. 1, No. 3; p. 20
11 = *Third-Party Software Development Systems*; Vol. 1, No. 1; P. 55

# The JOYS of COMPUTER GAMING



By Gary M. Kaplan

**It's A Dirty Job,**
**But Somebody's Gotta Do It . . .**

Over the last couple of months, I've had virtually no rest. First it was those pesky aliens: They hurled bombs, missiles, mines, and laser blasts at me around the clock. Some even tried to gobble me up on sight! No respect at all . . . These hordes of menacing foes must have come from nearly a dozen different hostile worlds. (Why is it I've never seen a "friendly" alien?) Each of these worlds evidently has its own individual concept of combat strategy, weapon design, ethics, and morality because the modes and severity of attacks differed widely. One thing, however, that all these dastardly devils had in common was their quarry—*me!*

Some of the attacking hordes were accompanied by a malevolent thumping as their precise marching formation advanced hypnotically toward my flimsy barricade. Others stayed stationary but hurled down torrents of lethal missiles that I had to alternately duck and target my lasers against. (My neighbors must have been really surprised when they noticed all that debris strewn across their yards . . . ) What? Was I nervous? Not too—that is, not until I had to pilot all those strange land and space craft—everything from X-wing fighters to futuristic prairie schooners. Just when I'd feel comfortable at the controls of one, **Ka-boom!**—I'd be under vicious attack, or **c-r-r-unch!**—smack in the middle of a deadly asteroid belt. Nothing like huge chunks of space rocks whizzing around your head to keep you on your toes . . .

But it didn't end with all those downright nasty aliens and slimy, vile space creatures. Oh no, not by a long shot. There were still the Empire forces to contend with. Here, however, the battles were more scattered and slower paced; I had time to launch torpedoes and probes, as well as assess casualty reports and plan long-term strategy. Just when everything was going so well, however, a Red Alert brought me back down to earth. It seems the "Cold War" was no longer so *cold* . . . and my country needed me to command a SAM (surface-to-air missile) site. With all that sophisticated RADAR equipment, it shouldn't have taken too long to finish "locking in" on the enemy missiles and blasting them to smithereens, but as fate would have it, another series of emergencies sent me packing—first to rescue a downed helicopter crew from shark-infested waters, and then on a hazardous journey to the moon and Mars, where I had to jockey my landing craft amidst some pretty rough terrain.

You'd think by then I should have received a few words of thanks, wouldn't you? But no, the moment I landed on Mars, some terrorists decided to have a field day . . . and there I was right back in the thick of things—commanding a bomb squad. Now defusing a time bomb is no Sunday picnic! If my nerves could withstand that heart-thumping activity, you'd think I'd be in pretty good shape for a few more adventures yet to come.

But nothing prepared me for what I was up against next. Certainly, no one told me when I took this job that hundreds of horrible deaths awaited just around the corner. All they talked about was the *treasure* and *glory!* But when I reached the edge of the high cliff, it was already too late to turn back: On my left, a hungry python slithered toward me; to the right, a quicksand bog surrounded by bleached bones awaited; and behind me, a large grizzly bear blocked my only path into the forest.

Well, luckily I got out of that one with my skin, but one adventure led to another . . . And before I could take some well-earned rest, I somehow had gotten involved in a pirate treasure hunt, an escape from an ancient pyramid, the ferreting out of an awesome secret on a savage island, saving a Count trapped by a fiendish curse, preventing a nuclear reactor from blowing its top, and alternately exploring a ghost town, mysterious fun house, ancient alien civilization, enchanted treasure world, and a dark kingdom populated by orcs, dwarves, an old dungeon master, a beautiful princess in distress, and an evil ringwraith.

Whew—I never thought that one mortal could get so tired. What I really needed was a chance to relax and unwind . . . So, handing my ticket to adventure over to Indiana Jones, I planned on doing nothing but sitting back in my favorite easy chair and listening to some good music. But *They* had other plans for me. It was no use complaining; I'd heard the argument a hundred times before: "It comes with the territory . . ." For some reason or other I was needed to run up a bankroll by betting on the ponies, lead a championship baseball team to victory, bring back a shiny bowling trophy, don my skiis to better the old slalom record, outrace a suicide car, and then to take part in a grueling decathlon.

After somehow getting through that long, long decathlon, I sat down to a nice big bowl of Wheaties and planned my R&R. Nothing too strenuous . . . nothing too mentally demanding . . . just some good clean fun. So off to the casinos for some baccarat, blackjack, craps, poker, and the slots. It was fun while it lasted, but *They* needed me back on the job again.

I knew something really BIG must have been in the works because of the way my training for the forthcoming mission was being carried out: plenty of practice with challenging word games, concentration exercises, and contortions with cantankerous colored cubes. *They* obviously wanted my wits sharp for the BIG assignment coming up. But before I'd find out what it was, there was an obstacle course to negotiate, and then the final test of my state of mental readiness —passage through a series of simply complex, complexly simple 3-D mazes. I almost didn't make it through that one . . .

Now I was ready. The BIG assignment finally came in: Someone was needed to guide a dumb chicken safely across a 20-lane highway . . . What? Enough is enough! Tell'em I'm not here. Guide a chicken across a road like that? Instant chicken salad—with me probably ending up being accused of fowl play . . . Let 'em get sombody else for that one. I wouldn't do it now even if *They* awarded me the Pullet-ser Prize!

## Micro Motivation Comes Full Circle

I've been sitting here now several hours thinking and wondering—thinking about those psychedelic-sounding escapades of mine, and wondering about the fantastic powers of imagination that we all must have—letting us see what we *want* or *expect* to see. In my case, it was easy because I had a partner—one who was, incidentally, a lot more patient than that dumb chicken I eventually got teamed up with. Who was this patient partner? Some gaunt guru? Or sinewy sorcerer? No, none of these. My partner in all this was a friendly Texan—a TI personal computer equipped with the latest in games software.

I never really cared very much for games. And even when it became obvious that microcomputers were rapidly becoming the ultimate "games machines," I still felt that all the excitement of video games was just a passing fancy. It was my belief that the popularity of *computer* games was simply due to people just trying to find additional uses for machines that they bought primarily for other purposes. Now I know better . . .

What we're now just starting to see happen is actually the *reverse*: This year, several million consumers will be considering *interactive* video (as opposed to *passive* TV watching)—games that they can play at home in the company of friends and family, instead of plunking their quarters down coin slots at crowded arcades or all-night grocery stores. When they start to shop around and compare prices and features, increasing numbers of them will start to find that the stand-alone, cartridge-based, dedicated games machines can be almost as much money as the new breed of lower-cost microcomputers.

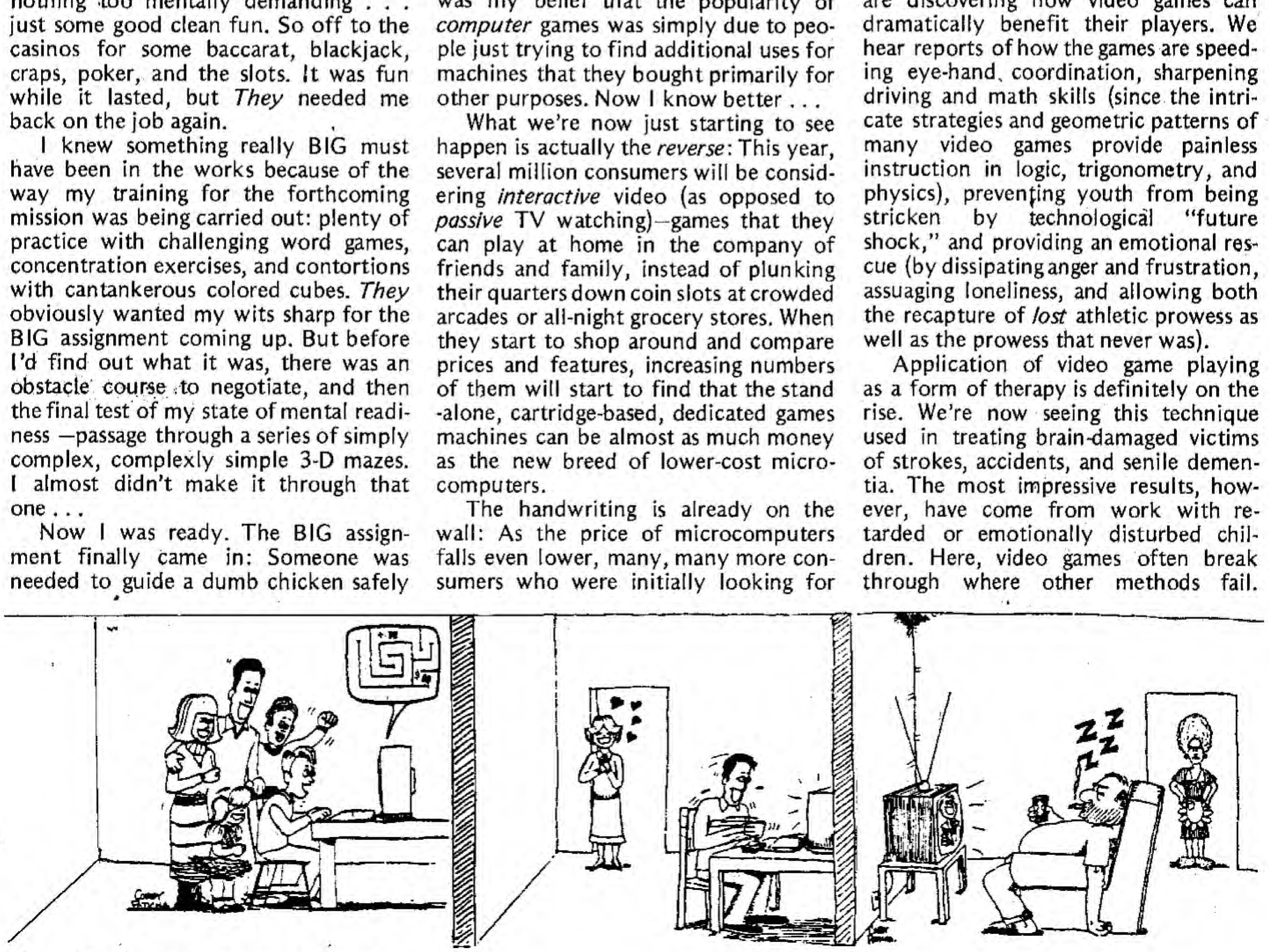
The handwriting is already on the wall: As the price of microcomputers falls even lower, many, many more consumers who were initially looking for video games machines will be able to justify the slightly higher cost of a full computer on the basis of potentially being able to do so much more than "just play games." Ironic, isn't it . . .

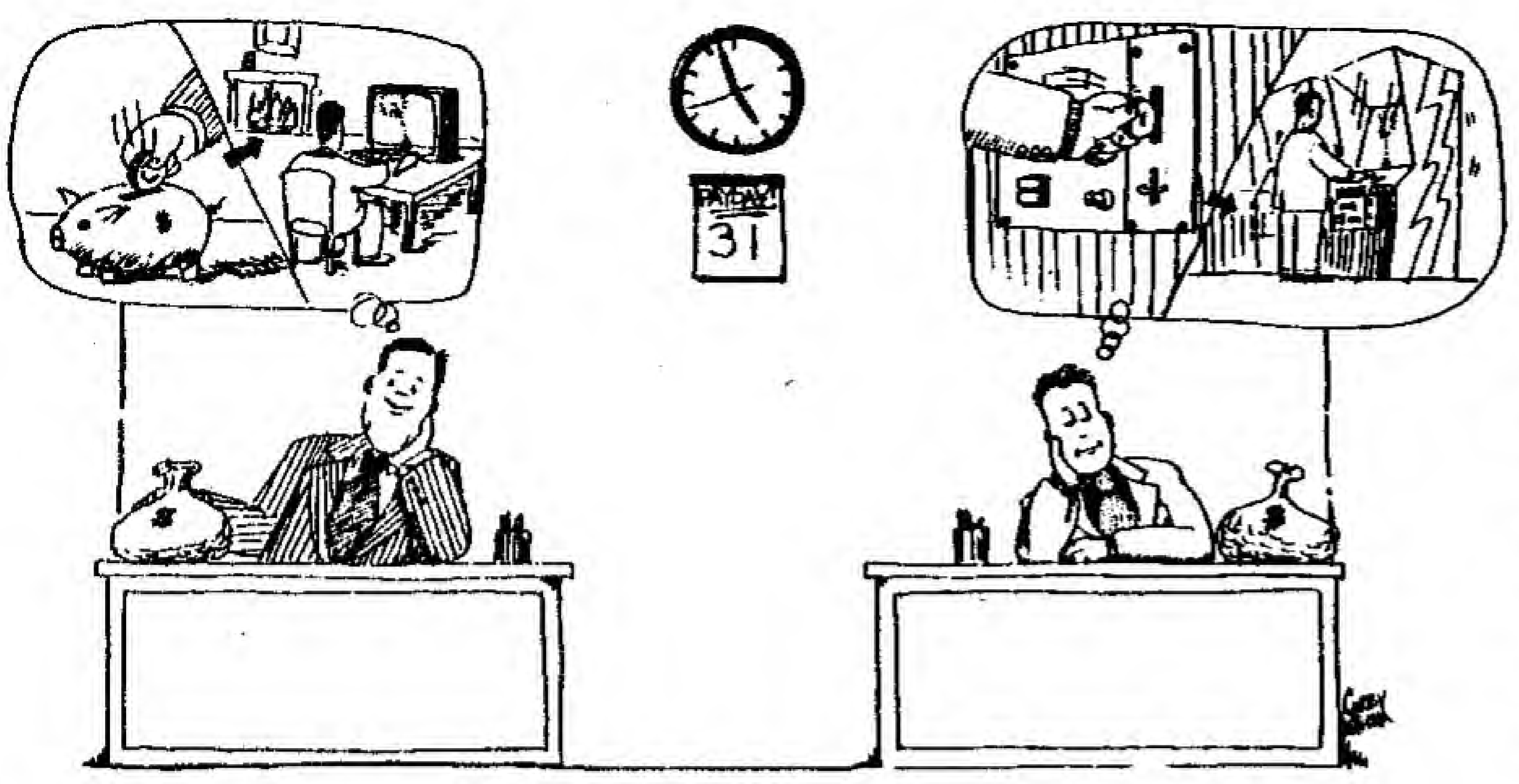
A few years ago, the great topic of speculation and cause for disagreement in the microcomputer industry was how to best increase public awareness and acceptance of these miracle machines so that a mass market with its lofty goal of "a computer in every home" could be eventually realized. Everything from electronic mail and banking, to education, home management, and tax/financial record keeping was nominated as being a likely candidate for the magic catalyst. Sure, entertainment was mentioned, but it was usually lumped together somewhat amorphously with home management and education. Nowhere do I remember anyone coming out and stating that it would be computer *gaming* that would ultimately be this catalyst and pave the way.

## The Seriousness of Playing Games

But regardless of whether video games are a primary or secondary motivation for getting a microcomputer, it's rapidly becoming obvious that electronic game playing isn't all just a game. Psychiatrists, psychologists, therapists, and educators are discovering how video games can dramatically benefit their players. We hear reports of how the games are speeding eye-hand coordination, sharpening driving and math skills (since the intricate strategies and geometric patterns of many video games provide painless instruction in logic, trigonometry, and physics), preventing youth from being stricken by technological "future shock," and providing an emotional rescue (by dissipating anger and frustration, assuaging loneliness, and allowing both the recapture of *lost* athletic prowess as well as the prowess that never was).

Application of video game playing as a form of therapy is definitely on the rise. We're now seeing this technique used in treating brain-damaged victims of strokes, accidents, and senile dementia. The most impressive results, however, have come from work with retarded or emotionally disturbed children. Here, video games often break through where other methods fail.





Psychologists have credited this to the "mastery experience" that is now possible for children who formerly were not able to be good in anything else. Until their exposure to games, they have never had a refuge of accomplishment from which to deal with the outside world. Once children become good at something (and, as a result, proud of their achievements), their attitudes and performance in other activities also dramatically improve.

## The Hardware Sets the Stage

As opposed to video games machines that are designed to be just "machines that play games," microcomputers are usually designed to perform *many* types of jobs, or handle certain types of work more efficiently. This architectural design determines the gaming environment that a particular computer will present to its users. Any limitations or constraints will be very obvious. For example, if a computer was designed without color graphics capability, then the games software compatible with this particular machine could not utilize color. Likewise, sound effects, music, 3-D animation, and synthetic speech are other game-enrichment capabilities that a microcomputer may or may not possess.

A comparison of all presently available microcomputers yields the surprising conclusion that *only* the Texas Instruments TI-99/4 and TI-99/4A personal computers have *all* the above named capabilities, and permit programmers to use them *all in the same program*. This represents an abundance of "raw materials" from which to construct games. When you combine this with TI's fast and powerful 16-bit microprocessor (TMS9900), it becomes apparent that these Texas Instruments personal computers offer one of the *best* gaming environments available.

The separate Video Display Processor chip (TMS9918A) that is inside the TI-99/4A is a good example of TI implementing internally in *hardware* what other computers require programmers to do in *software* (if indeed it can be done at all). This very sophisticated device gives the games programmer the ability to simply access and set in motion (independent of the program logic) 32 smoothly moving colored objects called "sprites"—objects whose shapes can very easily be defined, magnified, colored, given a 3-D overlapping appearance, and checked for collisions. These are the modular components from which many more exciting arcade-type games will be constructed in the future. [See next issue for a look at how the TMS9918A VDP chip operates.]

## A Little Software Magic

Although both Texas Instruments and third-party software developers got off to a relatively late and slow start in producing video games software, the present crop of new releases indicates that a massive catch-up campaign is in progress. The quality & quantity of the offerings that have recently crossed my desk in preparation for this article is truly impressive. Some of the highlights of what I experienced in my recent "quest for video understanding" deserve mention here:

First there were the arcade-type games. Texas Instruments has certainly come a long way since the early video games which Milton Bradley produced for them. *The Attack* is a case in point: At the time we originally reviewed it (May/June 1981), the game caused quite a sensation here because it was the first of its kind on the TI computer. It still is quite a hypnotic, exciting, and thoroughly enjoyable game, but when you compare its execution to the new crop of TI-produced arcade games, it leaves a lot to be desired. The most noticeable difference is in the joystick interaction and speed of response.

*Tombstone City* is a prime example of this difference. Although it is basically the same scenario—maneuver your craft and rid the screen of menacing creatures by blasting them to smithereens before they gobble you up—the speed at which you can maneuver and fight makes it an entirely different strategic game (one that starts to become extremely addictive if and when your skill progresses beyond a certain threshold). Although it's true that this particular game takes advantage of some TMS9900 Assembly Language coding which is faster than TI's special, more compact Graphics Programming Language (GPL) used in *The Attack*, the difference in interaction speed cannot be credited exclusively to the choice of language; rather, it appears that TI programmers have perfected their joystick interrogation software routines. It's *that* noticeable!

The scenarios of TI's other new video games are already familiar to the dedicated arcaders amongst you. There's just so much you can do with crash cars (*Carwars*), rows of marching aliens (*TI Invaders*), spinning tanks (*Blasto*), and dot-gobbling creatures that run through mazes (*Munchman*). The TI implementations, however, are noteworthy in that they don't simply try to *equal* the arcade versions, they attempt to *better* them.

The recent introduction of *TI Invaders* exemplifies this technical success. It will, I predict, be remembered in future years as the rallying point for a serious TI incursion into the home video games market. Virtually everyone who plays it will tell you that it is far better than any of the ubiquitous microcomputer versions, and even significantly more enjoyable and interesting than the original arcade machine that captured the attention (and quarters!) of games players everywhere.

As we go to press, it's too early to tell how *Munchman* will fare. At the recent Consumer Electronics Show [see related article in this issue], TI had a prototype up and running. It looked good, but not good enough, however, to impress the multitudes of "vidkids" already addicted to the *Pac-Man* arcade machine. (*All* home versions running on square picture tubes are typically at a disadvantage when trying to simulate the special visual effects of games on the custom, elongated tubes that are used with *Pac-Man* and other similar games.) In all fairness, it must be remembered that the version shown was only a prototype (disk-based), and I've heard recently from reliable sources that TI discovered a way to significantly improve the game prior to burning it into the Command Module cartridges. And if *TI Invaders* is any indication of what the Texas Instruments programmers are capable of, we all have something special to look forward to in the soon-to-be-released *Munchman* game.

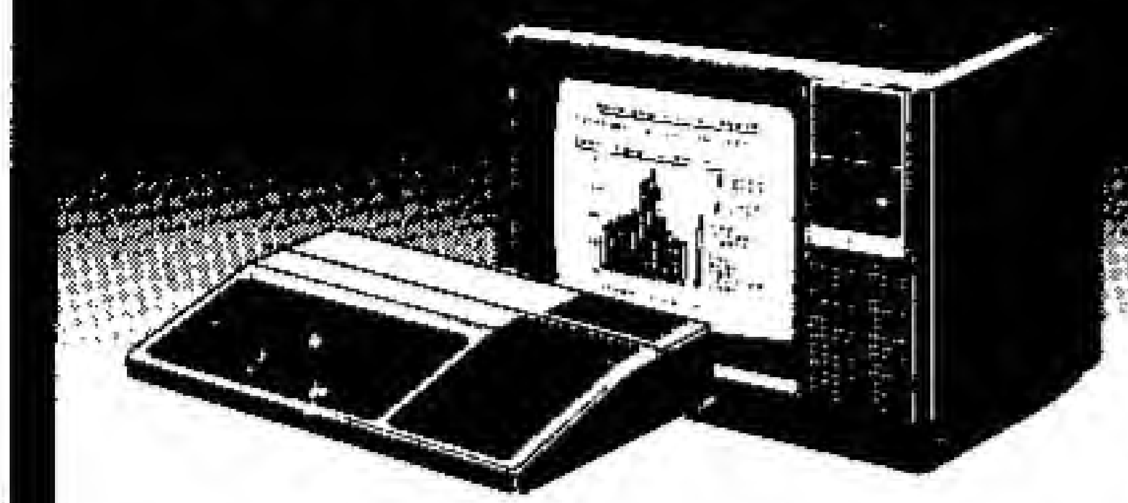
## Third-Party Players

Third-party software developers have also entered the arcade-game competition. I was impressed to see what could be done in Extended BASIC within the confines of the console's 16K memory by such firms as FFF Software (Trenton,

**TEXAS INSTRUMENTS**

**TI-99/4A**

**HOME COMPUTER**



Whether you're already knowledgeable about computers or want to learn how to get started, the TI-99/4A is for you. By simply snapping in one of the many Solid State Software Command Modules either you or just about anybody in your family can use the TI-99/4A because step-by-step instructions are displayed right on the screen. The TI-99/4A gives you an unmatched combination of features including:

- Powerful TI-BASIC
- Up to 72K total memory capacity
- NEW typewriter keyboard

Service is unparalleled with the broad TI nationwide service center network and TI's No Cost Exchange Warranty Program. The retail price is $550. Your special BACH Company price is a low **$369.50.** Below are just some of the accessories we have in stock for the TI-99/4A. Please call the number below to order or if you have any questions.

| | |
|---|---|
| Solid State Speech Synthesizer | **111.95** |
| Telephone Coupler | **167.95** |
| RS-232 Accessories Interface | **167.95** |
| Disk Drive Controller | **223.95** |
| Disk Memory Drive | **367.95** |
| Maxell MD1 Single Side 5¼" Floppy Disk. In Box of 10 Reg. Price 5.97ea | **2.91** |
| Maxell MD2 Double Side 5¼" Floppy Disk. In Box of 10 Reg. Price 6.97ea | **3.92** |
| Maxell FD2 Double Side 8" Floppy Disk. In Box of 10 Reg. Price 8.75ea | **5.97** |
| Solid State Printer | **296.95** |
| Memory Expansion (32K RAM) | **287.95** |
| R.F. Modulator (TV Adapter) | **37.95** |
| 10" Color Monitor | **327.95** |
| Wired Remote Controllers | **26.95** |
| Dual Cassette Cable | **11.95** |
| Monitor Cable | **14.95** |
| Thermal Paper | **7.95** |
| Physical Fitness | **21.95** |
| Teach Yourself BASIC | **26.95** |
| Weight Control and Nutrition | **44.95** |
| Extended BASIC | **74.95** |
| Personal Real Estate | **52.95** |
| Mailing List | **52.95** |
| The Attack Game | **29.95** |
| TI Invaders | **34.95** |
| Spellwriter | **24.95** |
| TI LOGO | **174.95** |
| Inventory Management | **52.95** |
| Invoice Management | **52.95** |
| Cash Management | **29.95** |

Business Packages Available—W/P, A/P, A/R, GL, P&L.

**ORDER TOLL FREE 800-227-8292**

California Residents call 415-494-1995. Use VISA, Mastercharge, check or money order. Indicate card number and expiration date. Order product #60. Add $4.50 for shipping.

**The BACH Company**
P.O. Box 51178
Palo Alto, CA 94303

NJ) with their *TI Asteroids* game, and the Extended Software Company (Cincinnati, OH) with *Gorfia Pestulitis*. These firms appear to be off to a real good start in arcade-style software production.

But space and combat action games don't always have to be arcade-style (simulated by the smoothly moving sprites accessible in Extended BASIC, Pascal, and Assembly Language) in order to be exciting and fun. Nowhere is this more evident than in the TI BASIC release from The Software Exchange Group (North Chili, NY), *Alien Attack* —a colorful game that deceptively starts off very slow and predictable, then continually throws little surprises at you as the challenge to your demonstrated skill intensifies.

Speaking of surprises, when I loaded up *SAM Defense* from the Futura Software line of Ehninger Associates (Fort Worth, TX) I was caught unprepared. Suspecting another *Star-Trek* clone, what I found instead was an extremely realistic simulation of an air defense radar-monitoring site (complete with realistic instrumentation). It was all the more impressive because of the enemy aircraft's jamming and elusive tactics capabilities built into the confines of the 16K BASIC memory.

The recent announcement that undoubtedly made many TI games players rejoice the most was the availability of the Scott Adam's *Adventure* series (contained in 11 separate TI packages) for the TI-99/4 and TI-99/4A. TI's hybrid firmware and software implementation makes a lot of sense: the user buys the master *Command Module* (and receives with it the first *Pirate Adventure* on cassette tape or disk), and then can add the different scenarios by purchasing the separate tapes or disks that all interact with the master module. This speeds up the games slightly, but most importantly, eliminates the need for any additional memory to run the programs (as is the case with implementations on other microcomputers), and makes it easy to save your progress on tape or disk.

Whereas the Scott Adam's Adventures are logical text-based puzzles that can leave you "terminally" frustrated if you can't surmount the individual puzzles and traps in a pre-determined sequence (even with the built-in "HELP" feature), the adventure games from Fantasy Computing (Escondido, CA) offer an entirely different type of challenge. With over 120 rooms to explore, *Ringwraith's Lair* (in Extended BASIC, requiring disk and 32K Memory Expansion) necessitates that the player keep a detailed map of his whereabouts to avoid getting lost (and to retrace his progress, picking up where he left off at each of the many sittings it will take to successfully complete this first scenario). The 25 monsters that lay lurking within the rooms should give even the most combat-crazy games player ample opportunity to satisfy his craving for a good fight.

I was also quite impressed with the interesting assortment of sports games that crossed my desk. Baseball was a big hit with me because of the Extended BASIC "double header" I was able to participate in—Futura's *All-Star Baseball*, and Extended Software Company's *Extended Baseball*. It's funny how *different* these two software packages really are —funny because you'd expect just the opposite. After all, the rules of the game are set in stone! But each program actually uses sprites to simulate a *different* aspect of the game, and uses the fixed color graphics and coding logic in a *different* way as well: The Futura version tests your fielding agility and the Extended version your hitting (with joysticks). I enjoyed both versions and only wish the developers could have gotten together and somehow produced a version with the best parts of both.

Extended BASIC certainly has added a new dimension to sports simulations, but by no means is the regular TI BASIC confined to the minor leagues. When in the hands of a master, it can be used to produce some sensational effects. I found this to be the case in the sports simulations from Pewterware (Gulf Breeze, FL). Both offerings make especially entertaining party games: *Bluegrass Sweepstakes* has all the unpredictability and excitement of a real horserace; *Decathlon* is a fine example of animation techniques determined by player input (a simple time estimate rather than a more realistic but memory-intensive moving graphics manipulation). It's remarkable how in each event the program can alternately make you feel great about and disappointed in your one-finger "athletic prowess."

When I started this marathon of computer gaming, the first thing I did was have my wife hide my Rubik's Cube so that there would be no distracting temptations from the job at hand. So guess what was waiting for me right on top of the software pile? That's right—none other than this mind-boggling mathematician's delight in *software* form! Linear Aesthetic Systems (West Cornwall, CT) has done quite a commendable job in bringing the pleasures and frustrations of *The Cube* to TI personal computer users. Their latest version (2.0) permits all of the same moves as the mechanical puzzle, plus provides commands for scrambling and unscrambling the cube, and a mode to save a scrambled cube on cassette tape. The cube itself is realistically simulated on the screen as a 3-D object with different colors for each face. If sorting through all the possible billions of combinations ever becomes too easy for you, don't worry . . . there's always *Quad Cube*—a brand new "twist" to the puzzle from the same firm.

Continued on p. 47

# ATTENTION 99/4 & 99/4A USERS

## International Home Computer Users Association

The **International Home Computer Users Association,** a multi-service, non-profit organization, has been established dedicated to serving the needs of Texas Instruments home computer users and users' groups.

The initial services offered are:

— An information and referral service
— A monthly newsletter
— A biweekly bulletin
— A newsletter exchange
— Special interest group coordination
— Club start-up kits and assistance
— An international amateur radio network of computers
— Consumer aid
— Programming assistance
— Hardware and software evaluations
— An annual convention
— A speakers bureau
— Special Users' Group Service

As the Association develops, more services will be added.

In the months ahead an advisory board will be formed comprised of TI Users' Group Presidents who will make proposals and suggestions to a full time staff and volunteers located at the Center in San Diego, California.

To participate in this Association the following affiliated memberships are offered. **INDIVIDUAL/FAMILY** — Open to individuals and families **$40, annually.**

**ASSOCIATED MEMBERSHIP** — Open to Users' Groups only **$65, annually.**

**SPONSORING MEMBERSHIP** — Open to individuals and other organizations wishing to support the Association **$250, annually.**

**CORPORATE MEMBERSHIP** — Open to all companies wishing to sustain the organization **$2,000, annually.**

Limited services are offered to non-profit, educational organizations without charge through the NPE Membership.

Don't be left out. Join ICA.
**International Home Computer Users' Association**
**P.O. Box 371**
**Rancho Santa Fe, CA 92067**

☐ **Individual Membership** I don't want to be left out. I want support!
☐ **Associated Membership** Users' Groups only. We want support!
☐ **Sponsoring Membership** We would like to support ICA!
☐ **Corporate Membership** We would like everyone to know that we support ICA.

Name ____________________

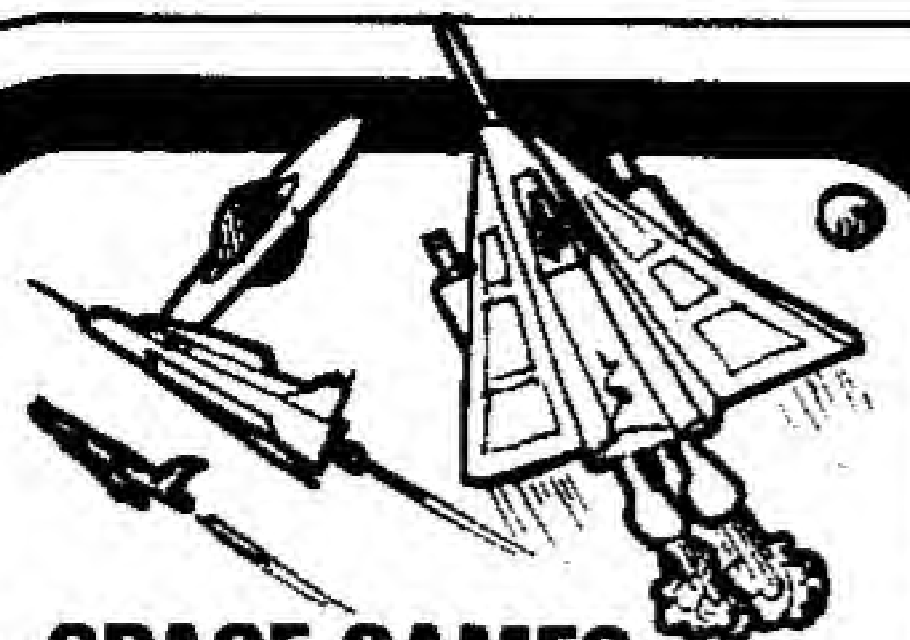
Address ____________________

City, State, Zip ____________________

Special Interests ____________________

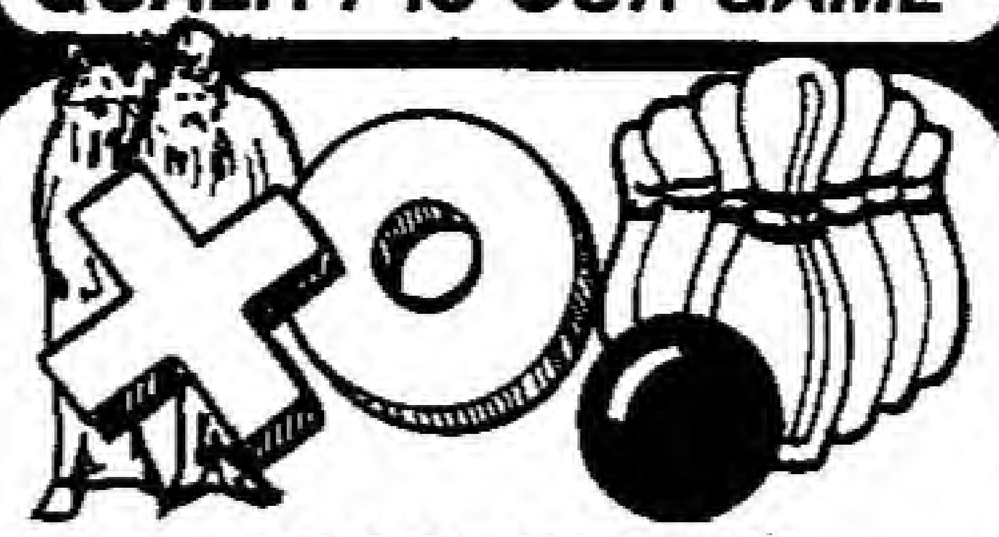
A membership package will be returned by mail!
**Please Enclose Payment**

"QUALITY IS OUR GAME"



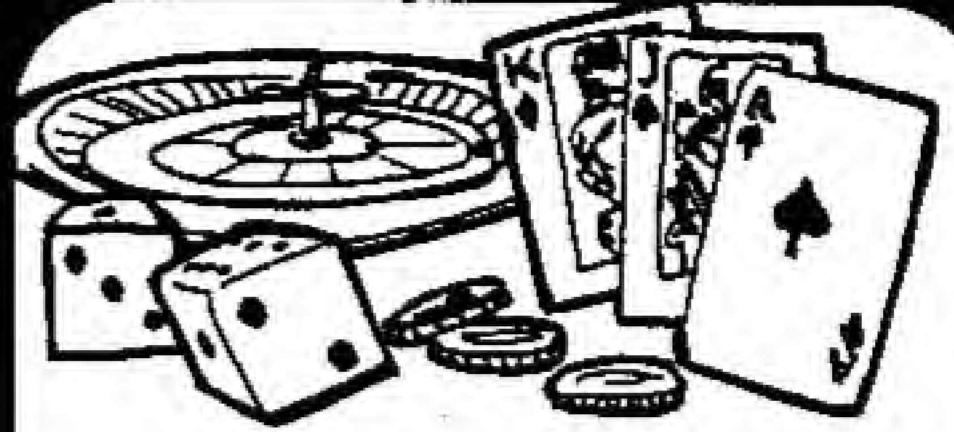
**SPACE GAMES VOL. 1**

- **ALIEN ATTACK** - 5 lives against the endless alien attack. Great action graphics.
- **INVADE THE EMPIRE** - Commanding a starship, must destroy four enemy vessels.
- **X-WING PILOT** - Squads of Ti-Fighters are escaping...destroy all or goodbye rebel.



**ASSORTMENT PACKAGE**

- **VIDEO BOWLING** - All the action of real bowling...auto-score, one or two players.
- **3-D TIC TAC TOW** - 3-D graphics, sound in this popular version of Tic Tac Toe
- **MONSTER MATCH** - Real memory challenge for two or three players, graphics, sound, music.



**ATLANTIC CITY**

- **SLOT MACHINE** - Supergraphic slot machine for one to three players.
- **BLACK JACK** - One or two players against the house, 52 card deck, graphics, sound.
- **SHIMMY** - One to three players in this exciting version of Baccarat, 52 card deck, graphics, sound.

**ORDERING INFORMATION:** Please enclose payment with order. Add $1.00 for postage. (Add $2.00 for postage on orders outside U.S.)

PROGRAMMED IN TI BASIC - 16K

CASSETTE ONLY $12.95
DISK ONLY $16.95

send check or money order to:

**the software exchange group**
P.O. Box 97
NORTH CHILI, N.Y. 14514
(716) 594-9474

**3 NEW RELEASES from**

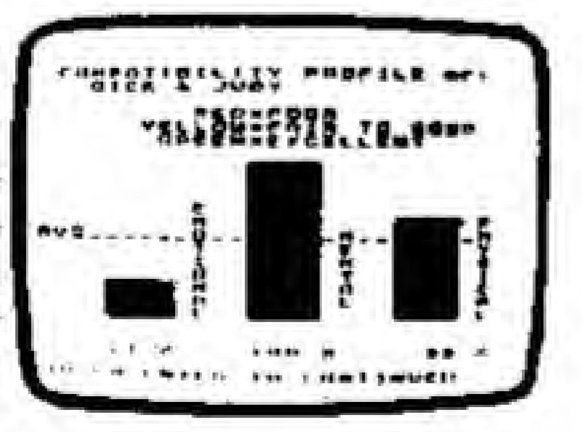
**PrP computergraphics**

All Programs
TI 99/4 CASSETTE FORMAT

**SCRIBBLE**

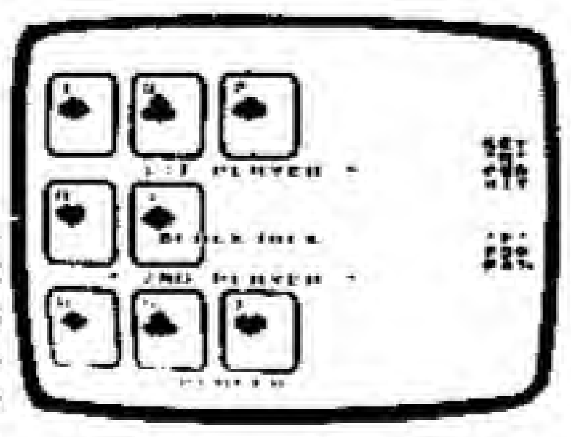
Fascinating computer word game. Players form cross words from 8 random letters offered by computer. Colorful Graphic playing board displayed. Scores are calculated and shown in personalized color Box Scores. Extra Bonus Color Squares in game. Exciting and easy to play. A PrP favorite.

**BIORYTHYM**



Outstanding use of Color Graphics show computer analysis of Bio Rhythms based on any birthdate since 1801. Displays mental, physical, and emotional status for any given day. Graphics dramatically change to show status of succeeding days. Same program contains a compatibility analysis for any two people. Results shown in spectacular color graphics with sound. Humorous comments. Exciting program from PrP.

**BLACKJACK**



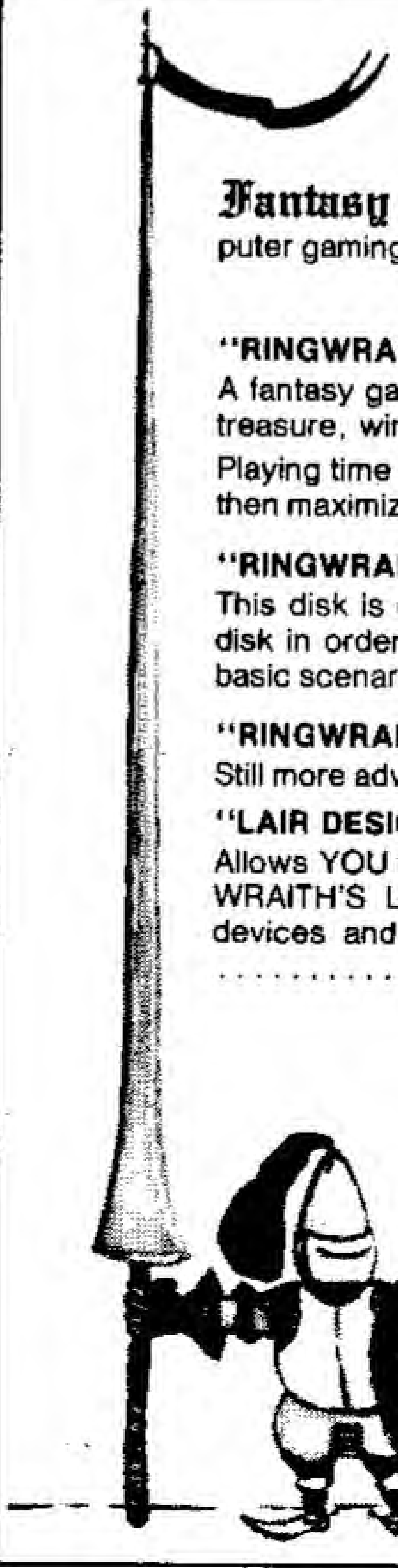
Computer, as Dealer, plays against 1 or 2 Players. Excellent Color Graphics actually display the dealt cards on the screen. Dealer's second card remains hidden until players pass or take hits. Up to 5 cards displayed for dealer and both players, all at the same time. Cards are dealt without repeats from 52 card deck. Fun, realistic. A PrP winner.

| | |
|---|---|
| Single Program | 7.95 each |
| Any 2 Programs | 6.95 each |
| Any 3 Programs | 5.95 each |

**PrP computergraphics**

Mail Check or Money Order To:

901 Cherrydale Street
Lake Charles, LA 70605



**Fantasy Computing** is dedicated to the idea that computer gaming should be challenging, fun and lasting in content.

***AVAILABLE NOW!***

**"RINGWRAITH'S LAIR"**
A fantasy gaming program in which the player tries to acquire treasure, win battles and survive to free a captured princess. Playing time varies with skill but cannot be mastered easily and then maximizing score will take much longer. . . . . . . **$24.95**

**"RINGWRAITH'S LAIR, SCENARIO II"**
This disk is used in conjunction with "RINGWRAITH'S LAIR" disk in order to create a new lair — totally different from the basic scenario. . . . . . . . . . . . . . . . . . **$12.00**

**"RINGWRAITH'S LAIR, SCENARIO III"**
Still more adventure in a new scenario! . . . . . . . . . . . **$12.00**

**"LAIR DESIGNER"**
Allows YOU to design and run your own "LAIR" within "RINGWRAITH'S LAIR" format. You design your own monsters, devices and location descriptions and store it all on disk. . . . . . . . . . . . . . . . . . . . . . . . . . . . . **$29.95**

*All of the above require TI-99/4 with disk, expansion memory, extended basic.*

**ORDER DIRECT**
**OR**
**WRITE FOR BROCHURE**

**Fantasy Computing**
1586 South Citrus
Escondido, CA 92027

## Texas Instruments TI-99/4A Home Computer Products

*Your Cost*

**CONSOLE**

| | | | |
|---|---|---|---|
| PHC | 004A | TI-99/4A Home Computer | 365.00 |

**PERIPHERALS**

| | | | |
|---|---|---|---|
| PHP | 1200 | Peripheral Expansion Box** | 194.00 |
| PHP | 1220 | RS-232 Card** | 136.00 |
| PHP | 1240 | Disk Controller Card** | 194.00 |
| PHP | 1250 | Expansion Box Disk Drive Card** | 299.00 |
| PHP | 1260 | Memory Expansion Card** (RAM) | 234.00 |
| PHP | 1270 | P-Code Card** | 194.00 |
| PHP | 1500 | Solid State Speech™ Synthesizer | 114.50 |
| PHP | 1600 | Telephone Coupler (Modem) | 168.50 |
| PHP | 1700 | RS-232 Accessories Interface[1] | 169.50 |
| PHP | 1800 | Disk Drive Controller[1] (One Disk Manager module packed with each Disk Controller) | 229.50 |
| PHP | 1850 | Disk Memory Drive | 368.50 |
| PHP | 1900 | Solid State Printer | 298.95 |
| PHP | 2200 | Memory Expansion[1] (32K RAM) | 288.95 |
| PHP | 2300 | VCR Controller[2]* | 569.95 |
| PHP | 2400 | P-Code Peripheral[1] | 314.00 |
| PHA | 2100 | R.F. Modulator (TV Adapter) | 38.95 |
| PHA | 4100 | 10" Color Monitor | 329.00 |

**OPTIONAL ACCESSORIES**

| | | | |
|---|---|---|---|
| PHP | 1100 | Wired Remote Controllers (Pair) | 28.50 |
| PHA | 1950 | Thermal Paper (2 Pack) | 9.50 |
| PHA | 2000 | Dual Cassette Cable | 12.50 |
| PHA | 2010 | Monitor Cable | 17.50 |
| PHA | 2020 | Audio Adapter (Headphone Jack) | 17.50 |
| PHA | 2310 | Panasonic VCR Controller Cable | 89.95 |
| PHA | 2320 | Sony VCR Controller Cable | 89.95 |
| PHA | 2330 | Pioneer VCR Controller Cable | 89.95 |
| PHA | 2605 | Blank Overlays (4 Pack) | 7.50 |

**DOCUMENTATION**

| | | | |
|---|---|---|---|
| PHA | 2600 | Beginning BASIC Manual (TI-99/4 Only) | 8.95 |
| PHA | 2601 | User's Reference Guide (TI-99/4 Only) | 8.95 |
| PHA | 2602 | Beginning BASIC Manual (TI-99/4A Only) | 8.95 |
| PHA | 2603 | User's Reference Guide (TI-99/4A Only) | 8.95 |
| PHA | 2606 | Creative Programming Computer Competency Series — Volume I | 8.95 |
| PHA | 2607 | Creative Programming Computer Competency Series — Volume II | 8.95 |
| PHA | 2608 | Creative Programming Computer Competency Series — Volume III | 8.95 |
| PHA | 2609 | Creative Programming Computer Competency Series — Allstar Projects | 8.95 |
| PHA | 2610 | Creative Programming Computer Competency Series (6 volumes) | 78.95 |

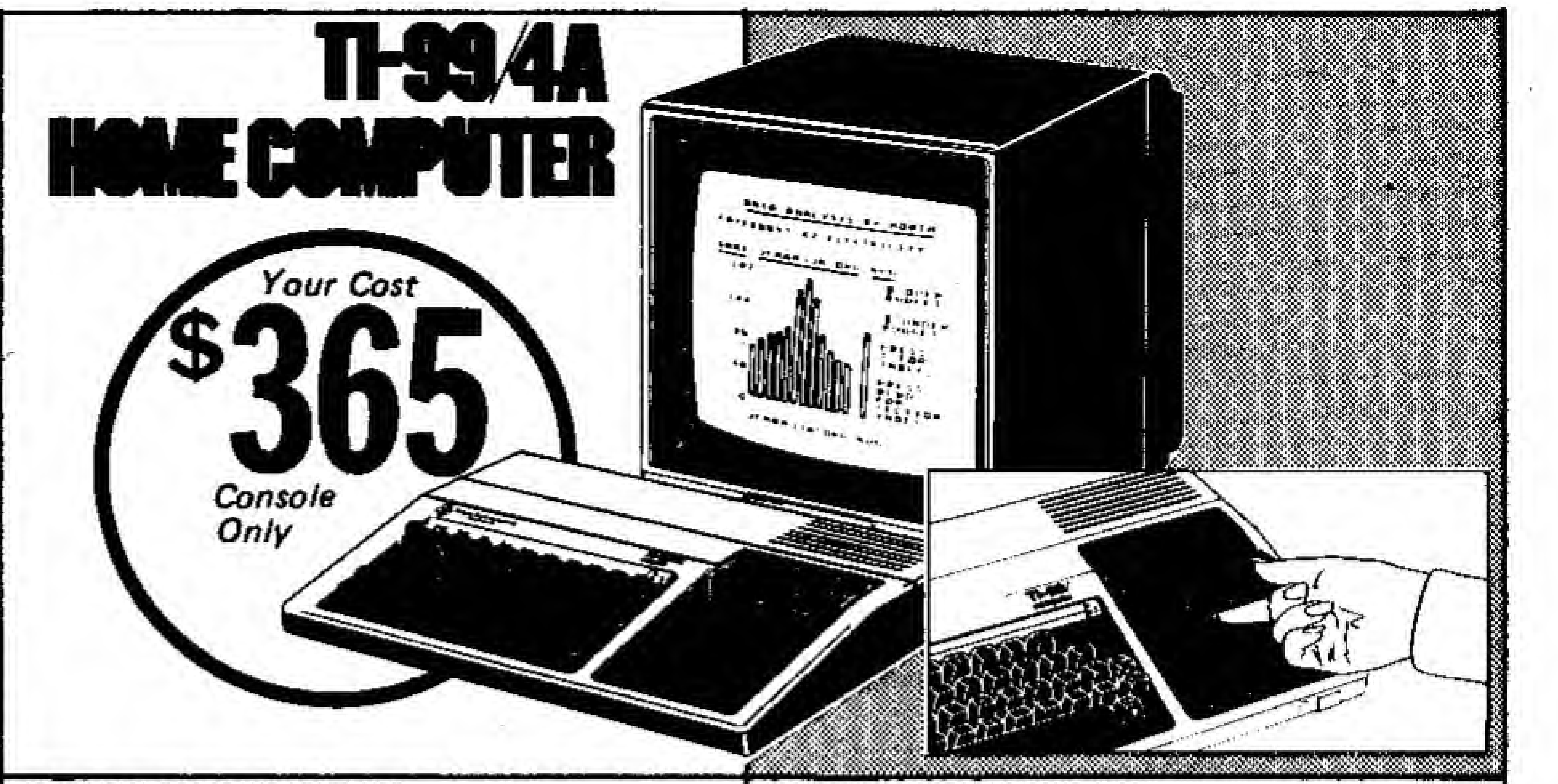
**APPLICATION PROGRAMS**

**Home Management/Personal Finance**

| | | | |
|---|---|---|---|
| | | **Command Modules** | |
| PHM | 3006 | Home Financial Decisions | 24.95 |
| PHM | 3007 | Household Budget Management (Data storage system is required) | 33.50 |
| PHM | 3012 | Securities Analysis | 45.95 |
| PHM | 3013 | Personal Record Keeping (Data storage system is recommended) | 39.50 |
| PHM | 3016 | Tax/Investment Record Keeping (Disk system is required) | 58.95 |
| PHM | 3022 | Personal Real Estate (Data storage system is recommended) | 58.95 |
| PHM | 3044 | Personal Report Generator (Data storage system is recommended) | 39.50 |
| | | **Diskette** | |
| PHD | 5001 | Mailing List | 54.50 |
| PHD | 5003 | Personal Financial Aids | 16.50 |
| PHD | 5021 | Checkbook Manager | 16.50 |
| PHD | 5022 | Business Aids Library — Finance Management (Extended BASIC Command Module is required) | 33.50 |
| PHD | 5024 | Business Aids Library — Inventory Management (Personal Record Keeping or Statistics Command Module is required) | 58.95 |
| PHD | 5027 | Business Aids Library — Invoice Management (Personal Record Keeping or Statistics Command Module is required) | 59.95 |
| PHD | 5029 | Business Aids Library — Cash Management (Extended BASIC Command Module is required) | 33.95 |
| PHD | 5038 | Business Aids Library — Lease/Purchase Decisions | 59.95 |
| | | **Cassette** | |
| PHT | 6003 | Personal Financial Aids | 12.50 |
| PHT | 6038 | Business Aids Library — Lease/Purchase Decisions | 54.95 |

**Education/Personal Enrichment**

| | | | |
|---|---|---|---|
| | | **Command Module** | |
| PHM | 3002 | Early Learning Fun | 24.95 |
| PHM | 3003 | Beginning Grammar | 24.95 |
| PHM | 3004 | Number Magic | 16.50 |
| PHM | 3005 | Video Graphs | 16.50 |
| PHM | 3008 | Video Chess | 54.50 |
| PHM | 3010 | Physical Fitness | 24.95 |
| PHM | 3015 | Early Reading † (Solid State Speech™ Synthesizer is required) | 45.95 |
| PHM | 3020 | Music Maker (Data storage system is recommended) | 33.50 |
| PHM | 3021 | Weight Control and Nutrition (Data storage system is recommended) | 49.95 |
| PHM | 3027 | Addition and Subtraction I† (Solid State Speech™ Synthesizer is recommended) | 33.50 |
| PHM | 3028 | Addition and Subtraction II† (Solid State Speech™ Synthesizer is recommended) | 33.50 |
| PHM | 3029 | Multiplication I† (Solid State Speech™ Synthesizer is recommended) | 33.50 |
| PHM | 3040 | TI LOGO (Memory Expansion is required) | 249.95 |
| PHM | 3043 | Reading Fun†* | 45.95 |
| PHM | 3059 | Scholastic Spelling — Level III** (Solid State Speech™ Synthesizer is required) | 45.95 |
| PHM | 3060 | Scholastic Spelling — Level IV** (Solid State Speech™ Synthesizer is required) | 45.95 |
| PHM | 3061 | Scholastic Spelling — Level V** (Solid State Speech™ Synthesizer is required) | 45.95 |
| PHM | 3062 | Scholastic Spelling — Level VI** (Solid State Speech™ Synthesizer is required) | 45.95 |
| PHM | 3064 | Typing Tutor** | 33.50 |
| | | **Diskette** | |
| PHD | 5007 | Teach Yourself BASIC | 28.95 |
| PHD | 5009 | Music Skills Trainer | 24.95 |



**APPLICATION PROGRAMS CONTINUED**

**Education/Personal Enrichment Continued** *Your Cost*

| | | | |
|---|---|---|---|
| | | **Diskette** | |
| PHD | 5011 | Computer Music Box | 16.50 |
| PHD | 5018 | Market Simulation | 16.50 |
| PHD | 5019 | Teach Yourself Extended BASIC (Extended BASIC Command Module is required) | 20.95 |
| PHD | 5020 | Music Maker Demonstration (Music Maker Command Module is required) | 12.50 |
| PHD | 5023 | Basketball Statistics (Extended BASIC Command Module is required) | 20.95 |
| PHD | 5026 | Bridge Bidding I | 24.95 |
| PHD | 5030 | Speak & Spell™ Program (Solid State Speech™ Synthesizer is required) | 24.95 |
| PHD | 5031 | Speak & Math™ Program (Solid State Speech™ Synthesizer and Terminal Emulator II are required) | 24.95 |
| PHD | 5039 | Bridge Bidding II | 24.95 |
| PHD | 5041 | Bridge Bidding III | 24.95 |
| PHD | 5042 | Spell Writer (Terminal Emulator II Command Module and Solid State Speech™ Synthesizer are required) | 24.95 |
| PHD | 5067 | Teach Yourself Beginning BASIC** | 24.95 |
| | | **Cassette** | |
| PHT | 6007 | Teach Yourself BASIC | 24.95 |
| PHT | 6009 | Music Skills Trainer | 20.95 |
| PHT | 6011 | Computer Music Box | 12.50 |
| PHT | 6018 | Market Simulation | 12.50 |
| PHT | 6019 | Teach Yourself Extended BASIC (Extended BASIC Command Module is required) | 16.60 |
| PHT | 6026 | Bridge Bidding I | 20.95 |
| PHT | 6031 | Speak & Math™ Program (Solid State Speech™ Synthesizer and Terminal Emulator II are required) | 20.95 |
| PHT | 6039 | Bridge Bidding II | 20.95 |
| PHT | 6041 | Bridge Bidding III | 20.95 |
| PHT | 6042 | Spell Writer (Solid State Speech™ Synthesizer and Terminal Emulator II Command Module are required) | 20.95 |
| PHT | 6067 | Teach Yourself Beginning BASIC** | 20.95 |

**Entertainment**

| | | | |
|---|---|---|---|
| | | **Command Module** | |
| PHM | 3009 | Football | 24.50 |
| PHM | 3018 | Video Games I | 24.50 |
| PHM | 3023 | Hunt the Wumpus | 20.95 |
| PHM | 3024 | Indoor Soccer | 24.50 |
| PHM | 3025 | Mind Challengers | 20.95 |
| PHM | 3030 | A-Maze-Ing | 20.95 |
| PHM | 3031 | The Attack†† | 32.95 |
| PHM | 3032 | Blasto†† | 20.95 |
| PHM | 3033 | Blackjack and Poker†† | 20.95 |
| PHM | 3034 | Hustle†† | 20.95 |
| PHM | 3036 | ZeroZap†† | 16.50 |
| PHM | 3037 | Hangman†† | 16.50 |
| PHM | 3038 | Connect Four†† | 16.50 |
| PHM | 3039 | Yahtzee†† | 20.95 |
| PHM | 3041D | Adventure (Pirate Adventure Diskette Game included) | 39.95 |
| PHM | 3041T | Adventure (Pirate Adventure Cassette Game included) | 39.95 |
| PHM | 3052 | Tombstone City: 21st Century | 32.95 |
| PHM | 3053 | TI Invaders | 32.95 |
| PHM | 3054 | Car Wars | 32.95 |
| PHM | 3057 | Munch Man** | 32.95 |
| | | **Diskette** | |
| PHD | 5002 | TI-Trek (with speech) | 12.50 |
| PHD | 5010 | Mystery Melody | 12.50 |
| PHD | 5015 | Oldies But Goodies — Games I | 16.50 |
| PHD | 5017 | Oldies But Goodies — Games II | 20.95 |
| PHD | 5025 | Saturday Night Bingo (Solid State Speech™ Synthesizer is required) | 24.95 |
| PHD | 5037 | Draw Poker | 20.95 |
| | | **Adventure Series** | |
| PHD | 5043 | Pirate Adventure | 24.95 |
| PHD | 5046 | Adventureland | 24.95 |
| PHD | 5047 | Mission Impossible | 24.95 |
| PHD | 5048 | Voodoo Castle | 24.95 |
| PHD | 5049 | The Count | 24.95 |
| PHD | 5050 | Strange Odyssey | 24.95 |
| PHD | 5051 | Mystery Fun House | 24.95 |
| PHD | 5052 | Pyramid of Doom | 24.95 |
| PHD | 5053 | Ghost Town | 24.95 |
| PHD | 5054 | Savage Island I & II | 32.95 |
| PHD | 5056 | Golden Voyage | 24.95 |

**APPLICATION PROGRAMS CONTINUED**

**Entertainment Continued** *Your Cost*

| | | | |
|---|---|---|---|
| | | **Cassette** | |
| PHT | 6010 | Mystery Melody | 7.95 |
| PHT | 6015 | Oldies But Goodies — Games I | 12.50 |
| PHT | 6017 | Oldies But Goodies — Games II | 16.50 |
| PHT | 6025 | Saturday Night Bingo (Solid State Speech™ Synthesizer is required) | 20.95 |
| PHT | 6037 | Draw Poker | 16.50 |
| | | **Adventure Series** | |
| PHT | 6043 | Pirate Adventure | 24.95 |
| PHT | 6046 | Adventureland | 24.95 |
| PHT | 6047 | Mission Impossible | 24.95 |
| PHT | 6048 | Voodoo Castle | 24.95 |
| PHT | 6049 | The Count | 24.95 |
| PHT | 6050 | Strange Odyssey | 24.95 |
| PHT | 6051 | Mystery Fun House | 24.95 |
| PHT | 6052 | Pyramid of Doom | 24.95 |
| PHT | 6053 | Ghost Town | 24.95 |
| PHT | 6054 | Savage Island I & II | 32.95 |
| PHT | 6056 | Golden Voyage | 24.95 |

**OTHER APPLICATION PROGRAMS**

| | | | |
|---|---|---|---|
| | | **Command Modules** | |
| PHM | 3000 | Diagnostic | 24.95 |
| PHM | 3001 | Demonstration | 58.95 |
| PHM | 3011 | Speech Editor (Solid State Speech™ Synthesizer is required) | 35.95 |
| PHM | 3014 | Statistics (Data storage system is recommended) | 35.95 |
| PHM | 3026 | Extended BASIC | 75.95 |
| PHM | 3035 | Terminal Emulator II | 39.50 |
| PHM | 3055 | Editor Assembler* | 89.95 |
| PHM | 3058 | Mini-Memory* | 89.95 |
| | | **Diskette** | |
| PHD | 5004 | Programming Aids I | 12.50 |
| PHD | 5005 | Programming Aids II | 20.95 |
| PHD | 5006 | Math Routine Library | 24.95 |
| PHD | 5008 | Electrical Engineering Library | 24.95 |
| PHD | 5012 | Programming Aids III | 16.50 |
| PHD | 5013 | Graphing Package | 16.50 |
| PHD | 5016 | Structural Engineering Library | 24.95 |
| PHD | 5044 | SMU Circuit Analysis I** | 124.95 |
| PHD | 5063 | UCSD-Pascal[3] Compiler** | 109.95 |
| PHD | 5064 | UCSD p-System[3] Assembler/Linker** | 89.95 |
| PHD | 5065 | UCSD p-System[3] Editor/Filer/Utilities** | 67.95 |
| PHD | 5066 | TI PILOT** | 69.95 |
| | | **Cassette** | |
| PHT | 6004 | Programming Aids I | 8.95 |
| PHT | 6006 | Math Routine Library | 20.95 |
| PHT | 6008 | Electrical Engineering Library | 20.95 |
| PHT | 6013 | Graphing Package | 12.50 |
| PHT | 6016 | Structural Engineering Library | 20.95 |
| PHT | 6044 | SMU Circuit Analysis I** | 124.95 |

**SOFTWARE LIBRARIES**

| | | | |
|---|---|---|---|
| PHL | 7001 | The Home Financial Manager | 119.95 |
| PHL | 7002 | The Family Entertainer | 74.95 |
| PHL | 7003 | The Elementary Educator | 89.95 |
| PHL | 7004 | The Music Educator | 56.95 |
| PHL | 7005 | The Super Programmer | 99.95 |
| PHL | 7006 | The Speaking Math Teacher | 99.95 |
| PHL | 7007 | The Speaking Reading Teacher | 92.95 |
| PHL | 7008 | The Speaking Scholastic Spelling Teacher** | 181.95 |
| PHL | 7009 | The TI Arcade Game Series | 99.95 |
| PHL | 7010 | The Milton Bradley Game Series | 94.95 |
| PHL | 7011 | The Computer Introductory Package | 99.95 |

*Available in First Quarter 1982.
**Available in Second Quarter 1982.
†Developed by Scott Foresman
††Developed by Milton Bradley — The Attack, Blasto, Hustle, ZeroZap, Connect Four and Yahtzee are trademarks of Milton Bradley.
[1]Available only until replaced by peripheral card.
[2]Pending FCC approval, available late First Quarter 1982.
[3]UCSD, UCSD Pascal and UCSD p System are all trademarks of the Regents of the University of California.

**Other products from Texas Instruments:**

| | |
|---|---|
| TI-59 | $179.95 |
| TI-58C | 92.95 |
| PC100C | 149.95 |
| Programmer (LCD-*new*) | 59.95 |

**OLYMPIC SALES COMPANY**

CABLE: "OLYRAV" LSA
**TELEX: 67 34 77**
Main Showroom & Offices:
216 So. Oxford Ave.
Los Angeles, CA 90004
WE HONOR VISA and MASTERCHARGE

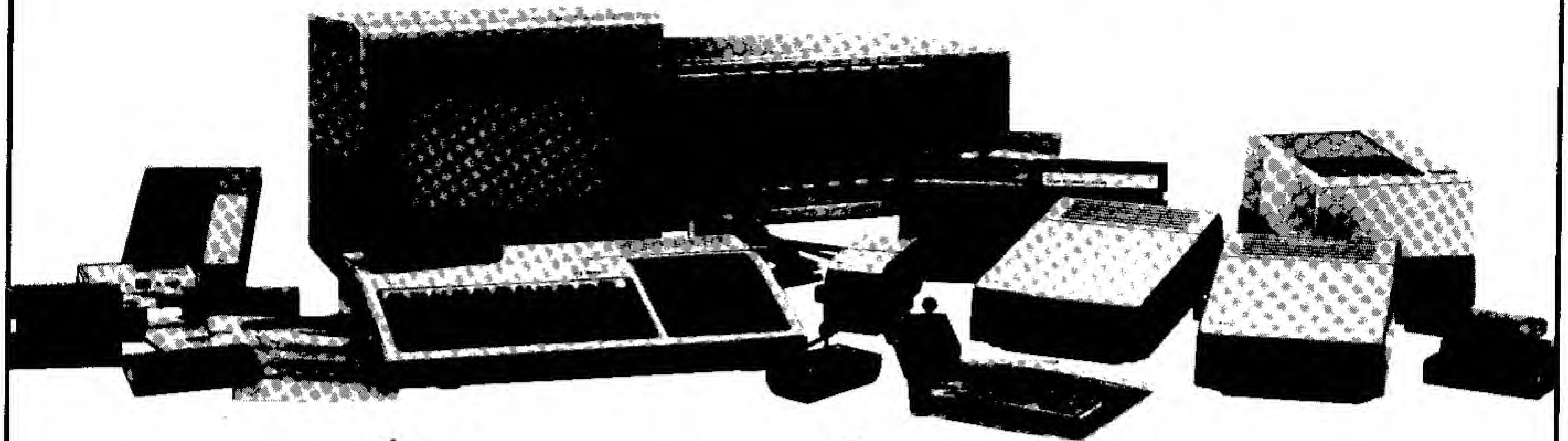
**ORDER DESKS open 7 Days a Week!**
7:00 AM to 7:00 PM Mon-Sat.
SUNDAYS NOON to 5:00 PM
Order Desks: (213) 739-1130

| TOLL-FREE (outside Calif.) | TOLL-FREE (within Calif.) |
|---|---|
| 800-421-8045 | 800-252-2153 |

*All goods subject to availability; this ad supersedes all previous ads; we are not responsible for typographical errors; we will meet or beat any advertised prices if the competition has the goods on hand.*
*Minimum shipping and handling $4.95.*
*All orders subject to verification and acceptance.*

*Send $1.00 in U.S./$5.95 foreign requests for our new 112-pg catalog-hot off the press!*

# Get any of these.



# At any of these.

DALLAS
422 **NorthPark Center**
Dallas, TX 75225
(214) 369-0444

**Prestonwood Town Center**
5301 Belt Line Rd. #1089
Dallas, TX 75240
(214) 386-0836

DENVER
E2282 **Cinderella City Shopping Center**
701 West Hamden Avenue
Englewood, CO 80154
(303) 781-5523

HOUSTON
3855 **Galleria Center**
Houston, TX 77056
(713) 961-5822

PALO ALTO
**Stanford Shopping Center**
71 Stanford Shopping Ctr.
Palo Alto, CA 94304
(415) 329-1784

PITTSBURGH
**Century III Mall**
3075 Clairton Rd. #412
West Mifflin, PA 15123
(412) 655-3700

SAN DIEGO
H3-3 **University Towne Center**
4465 La Jolla Village Drive
San Diego, CA 92122
(714) 452-7260

SAN FRANCISCO
100 **California Building**
Suite 120
San Francisco, CA 94111
(415) 392-5700

SANTA CLARA
**Marriott Industrial Park**
5353 Betsy Ross Drive
Santa Clara, CA
(408) 988-3481

SEATTLE
**Alderwood Mall**
3000 184th St. S.W. #580
Lynnwood, WA 98036
(206) 771-7111

ST. LOUIS
**Northwest Plaza Shopping Center**
782 Northwest Plaza
St. Ann, MO 73074
(314) 291-8262

THOUSAND OAKS
**The Oaks Shopping Center**
216 Hillcrest Avenue
Thousand Oaks, CA 91360
(805) 496-1717

Make the Texas Instruments Electronics Center your one-stop shopping source for all your TI Home Computer needs. Come in. Push buttons. Light lights. Run programs yourself or ask our expert staff of trained professionals for help. Choose from an exciting variety of software programs and peripheral equipment. It's all here. All in one place. It's a hands-on learning experience you won't want to pass up. Stop in or call our central toll-free number: **800-527-3503; 800-442-3522 in Texas.**

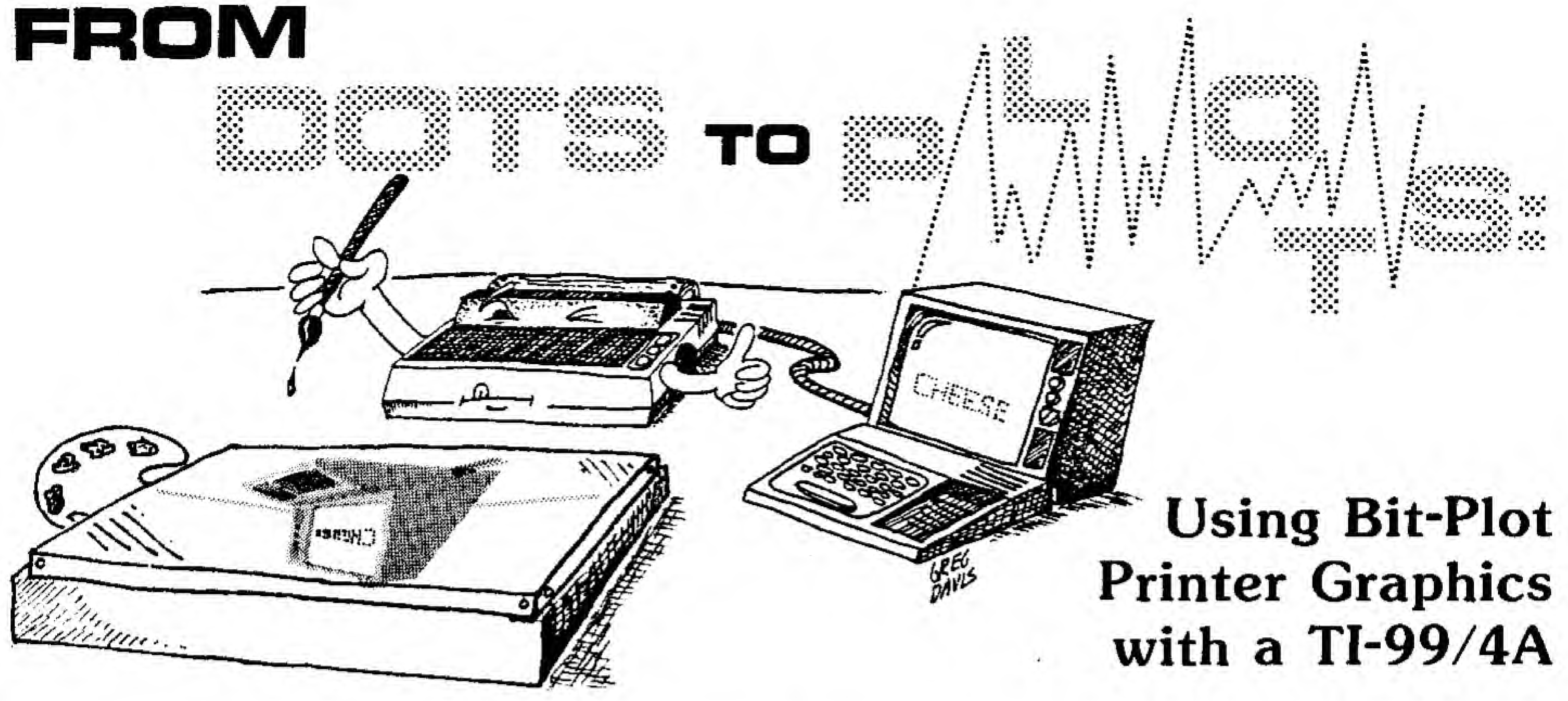
**Ask about our Special Software offer and our new Home Computer brochure.**

TEXAS INSTRUMENTS
ELECTRONICS CENTER

*Where all of TI's consumer products and accessories are available*

377398

# FROM DOTS TO PLOTS:



## Using Bit-Plot Printer Graphics with a TI-99/4A

By W.K. Balthrop

Back in our May/June, 1981 issue, we introduced you to the Epson MX-80 printer and showed you how to use its built-in block graphics for designs, business forms, and letterheads. Now, with the addition of Epson's new Graftrax 80 add-on kit, we can start to explore the world of "bit plot" printer graphics.

Before getting into an explanation of how bit image printing works, there are a few things you should know about the Graftrax 80 kit: (1) Besides the new graphics ability, Graftrax 80 also adds an italics print mode—thus doubling the number of print styles (from 12 to 24) that are available in normal text mode. (2) Installing the Graftrax 80 kit is fast and straightforward—requiring you to pull out one chip, plug three new ones into the empty sockets, cut one jumper on the printer's circuit board, and reset several DIP switches. (3) If you have the Epson model 8141 RS232 interface card, it won't work in the bit image mode; you'll need one of the cards with a 2K buffer—either the 8150 (which is being phased out) or the 8145.)

### Bit-Plot Graphics

In bit-image mode, the printer produces a character consisting of a column of from 0 to 8 dots at one time. This makes it possible to exactly duplicate the 8x8 pixel graphics characters of the TI-99/4 and 99/4A by printing 8 columns of up to 8 dots each on the printer. The printer dots are turned on in accordance with a binary format. For example, sending CHR$(0) to the printer will produce a blank space one dot-column wide; CHR$(1) will print only the bottom dot; CHR$(7) will print the bottom three dots; and CHR$(255) will print all 8 dots, and so forth (see Figure 1). Under software control you may select either 480 or 960 dot per line resolution. This means that to print a full line in the 960 dot mode you would have to print a dot-column character 960 times. For example:

```
10  FOR X = 1 TO 960
20  PRINT #1:CHR$(1)
30  NEXT X
40  END
```

This would print a line with only the bottom dot "on" across the entire page width. In the 960 mode, the line would appear solid with no space between dots. In the 480 mode, the line would have small but visible spaces between the dots.

[Note: There are two options in the 960 mode: the first at *half* the speed of the 480 mode, and the second at the *same* speed. This second mode may only be used by high-speed Assembly Language driver routines. BASIC interpreters are too slow in execution to print in this mode. If you try this mode using Extended BASIC you will loose many of the dots on each line. When using this high-speed mode, there is still another restriction: The same needle may not be struck twice in a row. The reason for this is that the needles take 2 micro seconds to hit and return to seat. Printing at 480 speed, the print head passes over a dot position every 1 microsecond. For this reason it is impossible to strike the same needle twice in a row at this high speed. If you attempt to strike the same needle twice in a row, the printer will automatically toss away the second consecutive dot. The printer will also print bi-directionally in this mode. It should be noted that there is some misalignment between passes of the printhead from opposite directions. This will vary from printer to printer and must be compensated for with computer software—Ed.]

To leave the Epson's standard text mode and enter the bit-image graphics mode, you must first send CHR$(27); "L" or "K";CHR$(X); CHR$(Y); to the printer. The ESCape "K" code will assign the 480 mode to the printer; "L" will assign it the 960 mode. You must then tell the printer how many graphics columns, or characters are to be printed. This is done with CHR$(X);CHR$(Y) where $0 \leq X \leq 255$, and $0 \leq Y \leq 3$. $(Y*256) + X$ = Number of columns of dots or characters to follow.

The only problem I have encountered with this convention is that it makes it dificult to intermix graphics and standard characters on the same line without complicated programming and file structures. The simplest method is to store a CHR$(X) in a file or data statement and then print CHR$(X) for each dot column across the page. This may be time consuming and require more disk, tape, or data space, but it allows for the least complicated program [and consequently the simplest example to get you started using this versatile graphics mode—Ed.].

The other thing that you will need to know to get the graphics working is how to format the OPEN statement. First, you must tell the RS232 port to output 8 bits instead of 7; then tell it to respond to the parity with **N** for No Parity. For example:

```
OPEN #1: "RS232.BA = 9600.DA = 8.PA = N"
```

### The Program

There are three main sections in the program. The first part of the program is a disk initialization subroutine. This routine will open a file on a blank disk with the following parameters: **RELATIVE**—random access of file record, and **INTERNAL, FIXED 24**—a fixed record lenth of 24 is used

**SAVE**

**Texas Instruments TI-99/4A**

| | |
|---|---|
| TI-99/4A Console | $379.00 |
| 10" Color Monitor | 329.00 |
| Disk Controller | 223.00 |
| Disk Memory Drive | 369.00 |
| Speech Synthesizer | 111.00 |
| RS232 Interface | 167.00 |
| Telephone Modem | 167.00 |
| RF Modulator | 37.50 |
| 32K Memory Expansion | 298.00 |
| Solid State Printer | 298.00 |
| Joysticks (pair) | 27.00 |

**Command Modules**

| | |
|---|---|
| Extended BASIC | 76.00 |
| TI LOGO | 209.95 |
| Terminal Emulator II | 39.95 |
| Household Budget Management | 34.95 |
| Personal Record Keeping | 39.95 |
| Video Chess | 54.95 |
| Football | 24.95 |
| Hunt the Wumpus | 21.95 |
| The Attack | 34.95 |
| Car Wars | 34.95 |
| TI Invaders | 34.95 |
| Tombstone City | 34.95 |
| Yahtzee | 20.95 |
| Blasto | 20.95 |
| Hustle | 20.95 |
| A-Maze-Ing | 20.95 |
| Connect Four | 16.95 |
| Addition & Subtraction 1 | 34.95 |
| Addition & Subtraction 2 | 34.95 |
| Multiplication 1 | 34.95 |
| Early Reading | 46.70 |
| Statistics | 37.95 |
| Diagnostic | 24.95 |

**Printers**

| | |
|---|---|
| TI 840RO Basic | 895.00 |
| TI 810RO Basic | 1480.00 |

Call for Prices
on other TI, Image,
Futura and Instant Software
for the TI 99/4 & TI 99/4A.

Prices FOB Lexington, KY
Add $4 for shipping
KY Residents add 5% tax
VISA &MC Orders add 3%

*ORDER TOLL FREE*
*(outside Kentucky)*
*1-800-354-9099*
*Kentucky Calls*
*606-276-1519*

**CBM INC.**
198 Moore Dr.
Lexington, KY 40503
1-800-354-9099

**ARE YOU LOOKING**

**FOR AN EXCITING JOB ?** ☐ NO ☐ YES

**Do you have experience with one or more of the following:** 9900 Assembly Language, Pascal, technical writing/editing, hardware design, magazine administration/production? ☐ NO ☐ YES

**Are you living in, or are willing to relocate** to the Pacific Northwest? ☐ NO ☐ YES

**Can you stand hard work,** deadlines, and get by with modest monetary compensation at first . . . ☐ NO ☐ YES

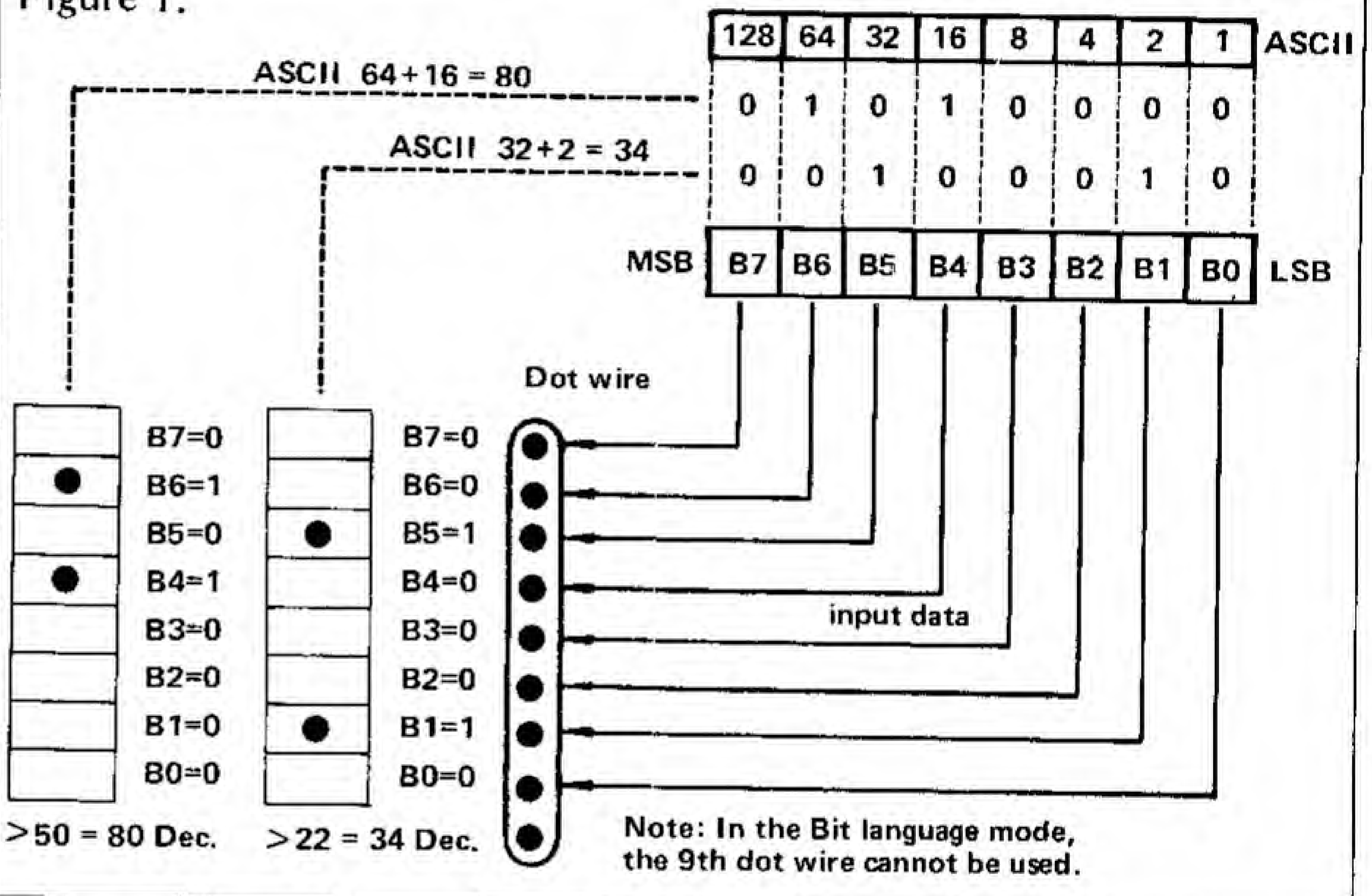
**Are you easy-going**, a non-smoker, and willing to work evenings, weekends, & holidays (if need be)? ☐ NO ☐ YES

**Do you want to work** with a group of people who answered YES to all the above, and put out a magazine just like this one? ☐ NO ☐ YES

**If you answered YES to all questions above, please reply in strictest confidence to:**

99'er Talent Hunt 2715 Terrace View Drive Eugene, OR 97405

Figure 1.



128 64 32 16 8 4 2 1 ASCII
ASCII 64+16 = 80
0 1 0 1 0 0 0 0
ASCII 32+2 = 34
0 0 1 0 0 0 1 0
MSB B7 B6 B5 B4 B3 B2 B1 B0 LSB
Dot wire
input data
B7=0 B6=1 B5=0 B4=1 B3=0 B2=0 B1=0 B0=0
B7=0 B6=0 B5=1 B4=0 B3=0 B2=0 B1=1 B0=0
>50 = 80 Dec.
>22 = 34 Dec.
Note: In the Bit language mode, the 9th dot wire cannot be used.

## ASCII Code Table

| | Hex. No. | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Hex. No | Binary No. | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 0 | 0000 | NUL 0 | 16 | SP 32 | 0 48 | @ 64 | P 80 | 96 | p 112 | NUL 128 | 144 | SP 160 | 0 176 | @ 192 | P 208 | 224 | p 240 |
| 1 | 0001 | 1 | DC1 17 | ! 33 | 1 49 | A 65 | Q 81 | a 97 | q 113 | 129 | DC1 145 | ! 161 | 1 177 | A 193 | Q 209 | a 225 | q 241 |
| 2 | 0010 | 2 | DC2 18 | " 34 | 2 50 | B 66 | R 82 | b 98 | r 114 | 130 | DC2 146 | " 162 | 2 178 | B 194 | R 210 | b 226 | r 242 |
| 3 | 0011 | 3 | DC3 19 | # 35 | 3 51 | C 67 | S 83 | c 99 | s 115 | 131 | DC3 147 | # 163 | 3 179 | C 195 | S 211 | c 227 | s 243 |
| 4 | 0100 | 4 | DC4 20 | $ 36 | 4 52 | D 68 | T 84 | d 100 | t 116 | 132 | DC4 148 | $ 164 | 4 180 | D 196 | T 212 | d 228 | t 244 |
| 5 | 0101 | 5 | 21 | % 37 | 5 53 | E 69 | U 85 | e 101 | u 117 | 133 | 149 | % 165 | 5 181 | E 197 | U 213 | e 229 | u 245 |
| 6 | 0110 | 6 | 22 | & 38 | 6 54 | F 70 | V 86 | f 102 | v 118 | 134 | 150 | & 166 | 6 182 | F 198 | V 214 | f 230 | v 246 |
| 7 | 0111 | BEL 7 | 23 | ' 39 | 7 55 | G 71 | W 87 | g 103 | w 119 | BEL 135 | 151 | ' 167 | 7 183 | G 199 | W 215 | g 231 | w 247 |
| 8 | 1000 | BS 8 | 24 | ( 40 | 8 56 | H 72 | X 88 | h 104 | x 120 | BS 136 | 152 | ( 168 | 8 184 | H 200 | X 216 | h 232 | x 248 |
| 9 | 1001 | HT 9 | 25 | ) 41 | 9 57 | I 73 | Y 89 | i 105 | y 121 | HT 137 | 153 | ) 169 | 9 185 | I 201 | Y 217 | i 233 | y 249 |
| A | 1010 | LF 10 | 26 | * 42 | : 58 | J 74 | Z 90 | j 106 | z 122 | LF 138 | 154 | * 170 | : 186 | J 202 | Z 218 | j 234 | z 250 |
| B | 1011 | VT 11 | ESC 27 | + 43 | ; 59 | K 75 | [ 91 | k 107 | { 123 | VT 139 | ESC 155 | + 171 | ; 187 | K 203 | [ 219 | k 235 | { 251 |
| C | 1100 | FF 12 | 28 | , 44 | < 60 | L 76 | \ 92 | l 108 | \| 124 | FF 140 | 156 | , 172 | < 188 | L 204 | \ 220 | l 236 | \| 252 |
| D | 1101 | CR 13 | 29 | - 45 | = 61 | M 77 | ] 93 | m 109 | } 125 | CR 141 | 157 | - 173 | = 189 | M 205 | ] 221 | m 237 | } 253 |
| E | 1110 | SO 14 | 30 | . 46 | > 62 | N 78 | ^ 94 | n 110 | ~ 126 | SO 142 | 158 | . 174 | > 190 | N 206 | ^ 222 | n 238 | ~ 254 |
| F | 1111 | SI 15 | 31 | / 47 | ? 63 | O 79 | _ 95 | o 111 | 127 | SI 143 | 159 | / 175 | ? 191 | O 207 | _ 223 | o 239 | 255 |

The heavy rules in the above table indicate special Epson printer control codes, and are not applicable to the discussion in the text.

to store 12 CHR$(X) values or 12 dot-columns of information.

It is possible to store up to 3798 such records on one 5¼" single-density disk. The process of initialization is therefore very slow and takes about one half hour. The initialize program will open the file and print CHR$(0) to all records. This is to help speed file building in the second section of the program. Where a large clear space on the paper is required, you do not need to enter all of these zeros.

The second section of the program is a form of "word processor," only here it is designed to handle *numbers* from 0 to 255. The program works with 20 file records at a time (240 character variables). Each group of 240 variables will be called one "created line." The present line being worked on is displayed at the top of the screen. The next value displayed is the position in that line, from 1 to 240. Below the position indicator is displayed the present CHR$ value at that position. Below that, the computer asks you for the new value that you want to assign to that position, from 0-255. Several single-keystroke commands are available to help you manipulate the data. By merely pressing ENTER without pressing any other key, the value of 0 will be assigned to the position indicated. The following is a list of commands and their explanations:

| | |
|---|---|
| P — | will print one line of data (240 dot positions of the line you are presently working on). |
| L — | will list all 240 variables on the screen for inspection. |
| N — | will allow you to jump to a new line number, and position—a process that would take too long by just using the arrow keys (below). Screen prompts will guide you. If you just hit ENTER, the program will default to the *current* line number or position without changing anything. |
| E — | will decrement the line number by 1. |
| X — | will increment the line number by 1. |
| S — | will decrement the position in a line by 1. |
| D — | will increment the position in a line by 1. |
| Z — | will return the user to the main menu screen. |

After entering a valid numeric value and pressing ENTER, the position in the line will automatically increment by 1 to the next record. Also, after entering the 240th record of a line, the line number will increment to the next line, and the position will return to number 1. The line will then also be automatically stored on disk. Note: Any time you change line numbers, the current line will also automatically be stored to disk. If you plan on exiting to the main menu or plan on turning the system off, you should first go to any other line so that the data are stored; otherwise the data on that line will be lost.

The final part of the program is the routine that prints all of your graphics. There are several options in this section. First is the option to print single density (480 dots per line), or double density (960 dots per line).

The second option is the line width: A line width of 240 will print one of the "created lines" in the create-file section, (240 dots); a line width of 480 will print 2 of your created lines per printed line (480 dots). To print 720 or 960 dots per line, you must first select the double density mode (960 dots per line). This will print 3 or 4 of your created lines according to the chosen parameters.

The last option is the number of lines you want printed: the number of lines you specify should be the number of actual lines which will appear on the paper, and not necessarily the number of "created lines." For example if you want to print 5 lines with 480 width, you will actually be printing 10 created lines.

| Print Line | "created lines" | | | |
|---|---|---|---|---|
| 1 — | Line 0 | Line 1 | | 5 printer lines = 10 "created lines" @ 480 width |
| 2 — | Line 2 | Line 3 | | |
| 3 — | Line 4 | Line 5 | | |
| 4 — | Line 6 | Line 7 | | |
| 5 — | Line 8 | Line 9 | | |
| 1 — | Line 0 | Line 1 | Line 2 | 3 printer lines = 9 "created lines" @ 720 width |
| 2 — | Line 3 | Line 4 | Line 5 | |
| 3 — | Line 6 | Line 7 | Line 8 | |

## EXPLANATION OF THE PROGRAM
*GRAFTRAX*

| Line Nos. | |
|---|---|
| 100-170 | REM |
| 180-190 | Initialize variables and array. |
| 210-290 | Data statements. |
| 300 | Subroutine to read data and display. |
| 310 | Subroutine to read data, display, and accept input. |
| 320 | Reads data only. |
| 330 | Initialize colors. |
| 340-380 | Checks for proper disk in drive #1. |
| 390-400 | Open file #2 on disk #1. |
| 410 | Clear screen with left to right scroll. |
| 420-430 | Control subroutine to check for proper disk in drive #1. |
| 440 | Reads multiple data statements and display. |
| 450-470 | Inputs record from disk #1. |
| 480-500 | Prints record on disk #1. |
| 510-550 | Improper disk in drive #1 messages; option to try again. |
| 560 | Opens a port to the printer if not already open. |
| 570 | Prints title page. |
| 580 | Prints option page and input option. |
| 590 | Checks input limits on options. |
| 600 | Branches to subroutines. |
| 610-1120 | Creates file subroutine. |
| 610-1120 | Creates file subroutine. |
| 610 | Checks for proper disk in drive #1. |
| 620 | Initializes variables. |
| 630 | Branches to subroutines. |
| 640-690 | Input option for density (480, 960). |
| 700 | Clear screen. Input first record. |
| 710-720 | Displays record variables. |
| 730 | Inputs new value or command. |
| 740-830 | Checks for a command input and does necessary logic. |
| 840-860 | Checks that all characters in the new value are numeric. |
| 870-920 | Assigns new value to array; advances to next line position, and checks for end of line. |
| 940-970 | Subroutine to enter a new line number and new position. |
| 980-1030 | Subroutine to print one line (240CHR$(X)). |
| 1040-1100 | Subroutine to display array contents on the screen. |
| 1110 | Subroutine to convert the input string into ASCII form and store it in the array. |
| 1120 | Subroutine to re-convert the array into a string for output to the disk. |
| 1130-1270 | Subroutine to print entire graphics page from information stored on disk. |
| 1130 | Checks for file on disk. |
| 1140 | Inputs density (480, 960). |
| 1150-1160 | Inputs width of graphics in dot columns. |
| 1170-1180 | Inputs number of lines to be printed. |
| 1190-1270 | Prints file from disk onto the printer in the form of bit image graphics. |
| 1280-1340 | Initializes a new disk with all CHR$(0); will destroy any records stored on that file. |
| 1350 | Closes files and ends. |

```
100 REM ************
110 REM * GRAFTRAX *
120 REM ************
130 REM
140 REM BY W. K. BALTHROP
150 REM 99'ER VERSION 1.4.1XB
160 REM
170 REM
180 OP2=0 :: OP3=0
190 DIM'Z(12,20)
200 GOTO 570
210 DATA 1,8,GRAFTRAX-80,3,8,SUBROUTINES,
    5,6,BY W.K. BALTHROP
```

# Fill this space with a GRAFTRAX graphic and win a trip to Japan.

## The Epson "Softwear" Sweepstakes.

We're looking for the Picasso of programming. So we drew up an art contest for people who don't know a painting pallet from a PROM.

If you've got an Epson printer, a computer and a little imagination, you could win a week-long trip for two to Japan. Or our top-of-the-line 136-column MX-100 printer. Or his and hers Seiko Quartz Watches. Or a whole lot of honorable mention prizes. And you'll get a T-shirt with the winning graphic just for entering.

All you have to do is program a GRAFTRAX graphic — abstract, landscape, still life, whatever — using an Epson MX-70, MX-80, MX-80 F/T or MX-100 printer. We'll not only put it on our T-shirts, we'll be displaying the winning entries for all to see in June at the National Computer Conference in Houston.

Why, you may ask, are we being so generous? It's simply because GRAFTRAX is the most incredible graphics capability made for micros. And we want to see it used to its full potential.

All entries will be judged on originality, creativity and best use of computer equipment. They must be postmarked no later than May 1, 1982, and be accompanied by the software program, so we can recreate the winning entries for verification. Make sure the graphic is no larger than 8" x 10" and no smaller than 4" x 6". And remember, if you digitize art or a photograph, it must have been originally created by you.

So get busy and enter. You might be a winner.

And your software could be your "softwear."

**EPSON**
EPSON AMERICA, INC.

3415 Kashiwa Street • Torrance, California 90505

## EPSON "SOFTWEAR" SWEEPSTAKES RULES

1) Any computer equipment may be used to format the entry, but the graphics output must have been printed on an Epson MX-70, MX-80, MX-80 F/T or MX-100 printer with either built-in or optional GRAFTRAX. Winning entries will be re-created by Epson for verification.

2) Each entry must be accompanied by the software program used to create it. All entries and software and the rights to use them become the property of Epson America, Inc.

3) All entries must be at least 6"x 4" and no larger than 8"x 10" in size.

4) Art or photographs, if used, must have been created by the entrant.

5) All entries will be judged by an independent panel of judges on their creative merit, originality and best use of computer equipment. Decision of the judges is final.

6) This contest is valid from January 1, 1982 until May 1, 1982. Entries must be postmarked no later than May 1, 1982.

7) Participation in the Epson "Softwear" Sweepstakes is open to any except the following: employees of Epson America, Inc., its service agencies, or their families.

8) Winners will be notified by mail no later than June 1, 1982. A list of winners will be made available by sending a stamped, self-addressed envelope to Epson America, Inc., 3415 Kashiwa Street, Torrance, CA 90505.

9) Entries will be maintained on file at Epson America, Inc. until January 1, 1983.

10) Prizes are as follows: First prize includes round-trip economy air transportation for two to Tokyo, from the airport nearest the winner's place of residence, and six nights standard hotel accommodations, double occupancy. Trip does not include airport departure taxes, hotel service charges, cost of transportation or other expenses incurred before leaving the airport of initial departure, returning to Tokyo airport and returning home from the airport of initial departure; nor does it include meals or gratuities. Second prize consists of one Epson MX-100 Printer. Third prize consists of his and hers Seiko Quartz Watches. Additional prizes include 25 Micro-Nine Printheads, 50 Epson Digital Watches, and 100 Epson Ribbon Cartridges.

11) You may enter more than once, but each entry must be accompanied by the official entry coupon below.

12) Void where prohibited by law.

- - - - - - - - - - - - - - - - - - - - - - -

Attach this form firmly to the back of each graphic you enter.

NAME ______________________________

STREET ______________________________

CITY ______________________________

STATE ______________ ZIP ______________

PHONE (______) ______________________________

COMPUTER EQUIPMENT USED

______________________________

______________________________

PRINTER MODEL AND SERIAL NUMBER

______________________________

______________________________

T-SHIRT SIZE __ S M L XL

Mail entries to:
"SOFTWEAR" SWEEPSTAKES
Epson America, Inc.
3415 Kashiwa Street
Torrance, California 90505

EPSON
EPSON AMERICA, INC.

## GRAFTRAX . . . from p. 19

```
220 DATA 1,11,MENUE,3,3,1.CREATE DATA FIELD,5,3,2.PRINT DATA
    FIELD,7,3,3.INITIALIZE NEW DISK,9,3,4.EXIT
230 DATA 1,5,CREATE DATA FIELD
240 DATA 1,9,PRINT DATA,3,3,FILE NAME?
250 DATA 23,3,YOUR CHOICE?
260 DATA 20,3,FILE NOT ON DISK
270 DATA 3,5,PLACE DISK IN DSK1,6,5,DISK IS NOT BLANK
280 DATA 24,3,PRESS ENTER TO CONTINUE
290 DATA 12,1,NEW LINE:,13,1,NEW POS.:,12,1,"",13,1,""
300 READ A1,A2,A$ :: DISPLAY AT(A1,A2):A$ :: RETURN
310 READ A1,A2,A$ :: DISPLAY AT(A1,A2):A$ :: ACCEPT AT(A1,A2+LEN(A$)+1):ANS$ :: RETURN
320 READ A1,A2,A$ :: RETURN
330 CALL SCREEN(5):: FOR A=1 TO 8 :: CALL COLOR(A,2,8):: NEXT A :: RETURN
340 OPEN #1:"DSK1.",RELATIVE,INTERNAL,INPUT
350 INPUT #1:A$,A,A,A
360 INPUT #1:A$,:: IF A$="" THEN CLOSE #1 :: GOTO 510
370 IF A$=ANS$ THEN CLOSE #1 :: GOTO 390 ELSE IF A$<>"" THEN 540
380 RESTORE 260 :: GOSUB 300 :: RETURN
390 IF OP2=1 THEN 400 :: OPEN #2:"DSK1."&ANS$,RELATIVE,INTERNAL,FIXED 24 :: OP2=1
400 RETURN
410 CALL COLOR(9,5,5):: CALL VCHAR(1,31,99,96):: CALL VCHAR(1,3,32,672):: RETURN
420 GOSUB 410 :: RESTORE 270 :: GOSUB 300 :: RESTORE 280 :: GOSUB 310
430 RESTORE 240 :: GOSUB 320 :: GOSUB 310 :: GOSUB 340 :: RETURN
440 FOR T1=1 TO T :: READ A1,A2,A$ :: DISPLAY AT(A1,A2):A$ :: NEXT T1 :: RETURN
450 FOR X=1 TO 20
460 INPUT #2,REC L*20+X:@$ :: GOSUB 1110
470 NEXT X :: RETURN
480 FOR X=1 TO 20
490 GOSUB 1120 :: PRINT #2,REC L*20+X:@$
500 NEXT X :: RETURN
510 IF PG=2 THEN GOSUB 390 :: RETURN ELSE DISPLAY AT(6,5):"DISK IS BLANK"
520 DISPLAY AT(7,1):"DO YOU WISH TO TRY AGAIN?    (Y/N)." :: ACCEPT AT(8,8)VALIDATE("YN"):AN$
530 IF AN$="Y" THEN 340 :: ELSE 1350
540 IF PG=1 THEN DISPLAY AT(6,1):"FILE NOT PRESENT" :: GOTO 520
550 DISPLAY AT(6,1):"DISK IS NOT BLANK" :: GOTO 520
560 IF OP3=1 THEN RETURN ELSE OPEN #3:"RS232.BA=9600.DA=8.PA=N" :: OP3=1 :: RETURN
570 GOSUB 330 :: GOSUB 410 :: RESTORE 210 :: T=3 :: GOSUB 440 :: RESTORE 280 ::
    GOSUB 310
580 GOSUB 410 :: RESTORE 220 :: T=5 :: GOSUB 440 :: RESTORE 250 :: GOSUB 310
590 IF ASC(ANS$)<48 OR ASC(ANS$)>55 THEN RESTORE 250 :: GOSUB 310 :: GOTO 590
600 ON VAL(ANS$)GOTO 610,1130,1280,1350
610 PG=1 :: GOSUB 410 :: RESTORE 230 :: GOSUB 300 :: GOSUB 420
620 L=0 :: P=1 :: P1=1
630 GOSUB 640 :: GOTO 700
640 DISPLAY AT(2,1):"WHAT DENSITY PER LINE?"
650 DISPLAY AT(3,1):"1. 480":"2. 960"
660 ACCEPT AT(5,1)VALIDATE("12"):D$ :: D=VAL(D$)
670 IF D=1 THEN D$="K" ELSE D$="L"
680 D=D*2
690 RETURN
700 GOSUB 410 :: GOSUB 450
710 DISPLAY AT(1,1):"LINE #:":"":"POSITION:":"":"OLD VALUE:":"":"NEW VALUE:"
720 DISPLAY AT(1,9):L :: DISPLAY AT(3,11):(P1-1)*12+P :: DISPLAY AT(5,12):Z(P,P1)
730 ACCEPT AT(7,12):NV$
740 IF NV$<>"S" THEN 770 ELSE IF P>1 THEN P=P-1 :: GOTO 720
750 P1=P1-1 :: P=12 :: IF L=0 THEN 720 ELSE IF P1<1 THEN GOSUB 480 :: L=L-1 :: P
    1=20 :: GOSUB 450 :: GOTO 720
760 GOSUB 450 :: GOTO 720
770 IF NV$="D" THEN IF P<12 THEN P=P+1 :: GOTO 720 ELSE P1=P1+1 :: P=1 :: IF P1>
    20 THEN GOSUB 480 :: L=L+1 :: P1=1 :: GOSUB 450 :: GOTO 32767
780 IF NV$="E" AND L>0 THEN GOSUB 480 :: L=L-1 :: GOSUB 450 :: P1=1 :: P=1 :: GOTO 720
790 IF NV$="X" AND L<100 THEN GOSUB 480 :: L=L+1 :: GOSUB 450 :: P1=1 :: P=1 :: GOTO 720
800 IF NV$="P" THEN GOSUB 560 :: GOSUB 980 :: GOTO 720
810 IF NV$="L" THEN GOSUB 1040 :: GOTO 710
820 IF NV$="Z" THEN 570
830 IF NV$="N" THEN GOSUB 940 :: GOSUB 300 :: GOSUB 300 :: GOTO 710
840 FOR TL=1 TO LEN(NV$)
850 IF ASC(SEG$(NV$,TL,1))<48 OR ASC(SEG$(NV$,TL,1))>57 THEN GOTO 720
860 NEXT TL
870 IF NV$="" THEN NV$="0"
880 NV=VAL(NV$):: IF NV<0 OR NV>255 THEN 720
890 Z(P,P1)=NV
900 P=P+1
910 IF P>12 THEN P1=P1+1 :: P=1 :: IF P1>20 AND L<100 THEN GOSUB 480 :: L=L+1 :: GOSUB 450 :
    : P1=1 :: P=1 :: GOTO 720
920 GOTO 720
930 GOTO 570
940 RESTORE 290 :: GOSUB 310 :: IF ANS$="" THEN NLINE=L :: GOTO 950 ELSE IF VAL(
    ANS$)<=189/D THEN NLINE=VAL(ANS$)ELSE GOTO 940
950 GOSUB 310 :: IF ANS$="" THEN LPOS=(P1-1)*12+P ELSE IF VAL(ANS$)<=240 THEN LP
    OS=VAL(ANS$)ELSE GOTO 950
960 IF NLINE>189/D OR NLINE<0 OR LPOS>240 OR LPOS<1 THEN 940
970 GOSUB 480 :: L=NLINE :: GOSUB 450 :: P1=INT((LPOS+1)/12):: P=((LPOS+1)/12-(P
    1*20))*12 :: RETURN
980 PRINT #3:CHR$(27);D$;CHR$(240);CHR$(0);
990 FOR X=1 TO 20
1000 PRINT #3:CHR$(Z(1,X));CHR$(Z(2,X));CHR$(Z(3,X));CHR$(Z(4,X));CHR$(Z(5,X));
     CHR$(Z(6,X));CHR$(Z(7,X));
1010 PRINT #3:CHR$(Z(8,X));CHR$(Z(9,X));CHR$(Z(10,X));CHR$(Z(11,X));CHR$(Z(12,X)
1020 NEXT X
1030 RETURN
1040 TB=1 :: LN=1 :: LI=9 :: GOSUB 1060 :: LN=10 :: LI=18 :: GOSUB 1060 :: LN=19 :: LI=20 :
     : GOSUB 1060
1050 GOSUB 410 :: RETURN
1060 FOR Y=LN TO LI
1070 FOR X=1 TO 12 :: PRINT TAB(TB*5-4);Z(X,Y);:: TB=TB+1 :: IF TB=6 THEN TB=1
1080 NEXT X :: NEXT Y :: PRINT :: PRINT
1090 RESTORE 280 :: GOSUB 310
1100 RETURN
1110 FOR @=1 TO 12 :: Z(@,X)=ASC(SEG$(@$,@,1)):: NEXT @ :: RETURN
1120 @$="" :: FOR @=1 TO 12 :: @$=@$&CHR$(Z(@,X)):: NEXT @ :: RETURN
1130 GOSUB 410 :: RESTORE 240 :: GOSUB 300 :: GOSUB 310 :: GOSUB 390
1140 GOSUB 640
1150 DISPLAY AT(7,1):"WIDTH OF GRAPHICS:":"1. 240":"2. 480":"3. 720 (960 RES. ONLY)":"4. 960
     (960 RES. ONLY)"
1160 ACCEPT AT(12,1)VALIDATE("1234"):WIDTH$ :: W=VAL(WIDTH$)
1170 DISPLAY AT(14,1):"NUMBER OF LINES? (1-";INT(189/W);")"
1180 ACCEPT AT(15,1)VALIDATE(DIGIT):NOL :: IF NOL>INT(189/W)THEN 1180
1190 DISPLAY AT(18,10):"PRINTING"
1200 GOSUB 560 :: PRINT #3:CHR$(27);"A";CHR$(8);
1210 FOR PRNT=0 TO NOL-1
1220 FOR PRIT=1 TO W
1230 PRINT #3:CHR$(27);D$;CHR$(240);CHR$(0);
1240 L=PRNT*W+PRIT-1 :: FOR X=1 TO 20 :: INPUT #2,REC L*20+X:@$ :: PRINT #3:@$;:: NEXT X
1250 NEXT PRIT :: PRINT #3:""
1260 NEXT PRNT
1270 GOTO 570
1280 GOSUB 410 :: DISPLAY AT(1,1):"INITIALIZATION WILL DESTROY":"ANY RECORDS BEING STORED":"ON
     THE FILE"
1290 DISPLAY AT(4,1):"DO YOU WISH TO CONTINUE?":"(Y/N)"
1300 ACCEPT AT(5,7)VALIDATE("YN"):AN$ :: IF AN$="N" THEN 570
1310 PG=2 :: GOSUB 410 :: DISPLAY AT(2,1):"PLACE BLANK DISK IN DRIVE #1" :: RESTORE 280 :
     : GOSUB 310
1320 GOSUB 430
1330 FOR X=1 TO 12 :: @$=@$&CHR$(0):: NEXT X
1340 FOR X=1 TO 3798 :: PRINT #3:@$ :: NEXT X :: CLOSE #2 :: GOTO 570
1350 CLOSE #2 :: CLOSE #3 :: END
```

# Now! Get the TI software you've always wanted FREE when you buy TI-99/4 software or peripherals from Unisource Electronics. It's simple. Read on!

The Specials listed below are designed to give you FREE Software for your TI-99/4 depending on the dollar amount of software or peripherals you buy from this ad. For example, if you buy $50 worth of software shown on the facing page, you can select any one of the items listed in Box 4. If you happen to want more software (or peripherals) — let's say $400 worth — you could select two items from Box 1 and one from Box 2, four items from Box 2, or any other similar combination. The choice is yours. In this example you could receive up to $120 in FREE Software.

## SPECIAL #1

When you buy $150 or more worth of software or peripherals, you get your choice of any item in Box 1 FREE.
Values of up to $40!

**BOX #1**

- TI Invaders (Module)
- Tombstone City (Module)
- Car Wars (Module)
- Household Budget Management (Module)
- Business Aids - Finance Mgmt. (Diskette)
- Music Maker - (Module)
- Addition & Subtraction I (Module)
- Addition & Subtraction II (Module)
- Multiplication 1 (Module)
- Teach Yourself Basic (Diskette)
- The Attack (Module)
- Remote Controllers (Accessory)
- Household Inventory System (Diskette)

## SPECIAL #2

When you buy $100 to $150 worth of software or peripherals, you get your choice of any item in Box 2 FREE.
Values of up to $30!

**BOX #2**

- Your choice of Adventure Series (Disc./Cass.)
- Early Learning Fun (Module)
- Beginning Grammar (Module)
- Home Financial Decisions (Module)
- Music Skills Trainer (Diskette or Cassette)
- Teach Yourself Extended Basic (Diskette)
- Bridge Bidding I (Diskette or Cassette)
- Bridge Bidding II (Diskette or Cassette)
- Bridge Bidding III (Diskette or Cassette)
- Teach Yourself Basic (Cassette)
- Math Routine Library (Diskette or Cassette)
- Electrical Eng. Lib. (Diskette or Cassette)
- Football (Module)
- Video Games (Module)
- Hunt The Wumpus (Module)
- Indoor Soccer (Module)
- Mind Challengers (Module)
- A-Maze-ing (Module)
- Blasto (Module)
- Oldies But Goodies II (Diskette)
- Blackjack & Poker (Module)
- Hustle (Module)
- Programming Aids II (Diskette)
- Structural Eng. (Diskette or Cassette)

## SPECIAL #3

When you buy $75 to $100 worth of software or peripherals, you get your choice of any item in Box 3 FREE.
Values of up to $22!

**BOX #3**

- Personal Financial Aids (Diskette)
- Checkbook Manager (Diskette)
- Number Magic (Module)
- Computer Music Box (Diskette)
- Market Simulation (Diskette)
- Teach Yourself Extended Basic (Cassette)
- Zero Zap (Module)
- Hangman (Module)
- Connect Four (Module)
- Oldies But Goodies I (Diskette)
- Oldies But Goodies II (Cassette)
- Draw Poker (Cassette)
- Programming Aids III (Diskette)
- Graphing Package (Diskette)
- All Star Baseball (Cassette)
- All Star Bowling (Cassette)
- Casino Pack (Diskette)
- Challenge II (Cassette)
- Chutes & Sharks (Cassette)
- Dr. Nuttier (Diskette)
- Gallactic War (Diskette or Cassette)
- Leapfrog (Cassette)
- Sam Defense (Cassette)
- Wall Street (Cassette)

## SPECIAL #4

When you buy $50 to $75 worth of software or peripherals, you get your choice of any item in Box 4 FREE.
Values of up to $15!

**BOX #4**

- Personal Financial Aids (Cassette)
- Computer Music Box (Cassette)
- Market Simulation (Cassette)
- TI - Trek (Diskette)
- Mystery Melody (Diskette or Cassette)
- Oldies But Goodies I (Cassette)
- Programming Aids I (Disk./Cass.)
- Graphing Package (Cassette)
- Dual Cassette Cable (Accessory)
- Thermal Paper (For Printer)

**Choose the items you want from the facing page, select your FREE merchandise from the specials above, and call the Unisource Hotline to place your order: 1-800-858-4580 (In Texas 1-806-745-8835).**

## Choose from this extensive selection of Software. NEW offerings from Unisource are marked with a dot.

| Package Title | Command Module | Diskette | Cassette |
|---|---|---|---|
| **Home Management Personal Finance** | | | |
| Home Financial Decisions | $ 29.95 | — | — |
| Household Budget Management | 39.95 | — | — |
| Securities Analysis | 54.95 | — | — |
| Personal Record Keeping | 49.95 | — | — |
| Tax/Investment Record Keeping | 69.95 | — | — |
| Personal Real Estate | 69.95 | — | — |
| •Personal Report Generator | 49.95 | — | — |
| •Mailing List * | — | 49.95 | — |
| Personal Financial Aids | — | 19.95 | — |
| Checkbook Manager | — | 19.95 | — |
| Bus. Aids Lib. - Inventory Mgmt | — | 69.95 | — |
| Bus. Aids Lib. - Invoice Mgmt | — | 69.95 | — |
| Bus. Aids Lib. - Cash Mgmt | — | 39.95 | — |
| •Bus. Aids Lib. Finance Mgmt | — | 39.95 | — |
| •Bus. Aids Lib. Lease/Purchase Decisions | — | 69.95 | 59.95 |
| Personal Financial Aids | — | — | 14.95 |
| •Accounts Payable * | — | 99.95 | — |
| •Accounts Receivable * | — | 99.95 | — |
| •General Ledger * | — | 99.95 | — |
| •Household Inventory System * | — | 34.95 | — |
| •Inventory Management * | — | 99.95 | — |
| •Mailing List * | — | 49.95 | — |
| •Payroll * | — | 99.95 | — |
| **Education/Personal Enrichment** | | | |
| Early Learning Fun | 29.95 | — | — |
| Beginning Grammar | 29.95 | — | — |
| Number Magic | 19.95 | — | — |
| Video Graphs | 19.95 | — | — |
| Video Chess | 69.95 | — | — |
| Physical Fitness | 29.95 | — | — |
| Early Reading † | 54.95 | — | — |
| Music Maker | 39.95 | — | — |
| Weight Control and Nutrition | 59.95 | — | — |
| Addition and Subtraction I † | 39.95 | — | — |
| Addition and Subtraction II † | 39.95 | — | — |
| Multiplication I † | 39.95 | — | — |
| TI LOGO (Requires Mem. Exp. Unit) | 224.95 | — | — |
| •Reading Fun † | 54.95 | — | — |
| Teach Yourself BASIC | — | 34.95 | 29.95 |
| Music Skills Trainer | — | 29.95 | 24.95 |
| Computer Music Box | — | 19.95 | 14.95 |
| Market Simulation | — | 19.95 | 14.95 |
| Teach Yourself Extended BASIC | — | 24.95 | 19.95 |
| Music Maker Demonstration | — | 14.95 | — |
| •Basketball Statistics | — | 24.95 | — |
| Bridge Bidding I | — | 29.95 | 24.95 |
| Speak & Spell Program | — | 29.95 | — |
| Speak & Math Program | — | 29.95 | 24.95 |
| Bridge Bidding II | — | 29.95 | 24.95 |
| Bridge Bidding III | — | 29.95 | 24.95 |
| Spell Writer | — | 29.95 | 24.95 |
| **Entertainment** | | | |
| Football | 29.95 | — | — |
| Video Games I | 29.95 | — | — |
| Hunt The Wumpus | 24.95 | — | — |
| Indoor Soccer | 29.95 | — | — |
| Mind Challengers | 24.95 | — | — |
| A-Maz-ing | 24.95 | — | — |

| Package Title | Command Module | Diskette | Cassette |
|---|---|---|---|
| The Attack †† | 39.95 | — | — |
| Blasto †† | 24.95 | — | — |
| Blackjack and Poker | 24.95 | — | — |
| Hustle †† | 24.95 | — | — |
| Zero Zap †† | 19.95 | — | — |
| Hangman †† | 19.95 | — | — |
| Connect Four †† | 19.95 | — | — |
| Yahtzee †† | 24.95 | — | — |
| •Adventure (Pirate Adventure Diskette Game included) | 49.95 | — | — |
| •Adventure (Pirate Adventure Cassette Game Included) | 49.95 | — | — |
| •Tombstone City: 21st Century | 39.95 | — | — |
| •TI Invaders | 39.95 | — | — |
| •Car Wars | 39.95 | — | — |
| •Munch Man | 39.95 | — | — |
| TI-Trek (w/speech) | — | 14.95 | — |
| Mystery Melody | — | 14.95 | 9.95 |
| Oldies But Goodies - Games I | — | 19.95 | 14.95 |
| Oldies But Goodies - Games II | — | 24.95 | 19.95 |
| Saturday Night Bingo | — | 29.95 | 24.95 |
| Draw Poker | — | 24.95 | 19.95 |
| **Adventure Series** | | | |
| •Pirate Adventure | — | 29.95 | 29.95 |
| •Adventureland | — | 29.95 | 29.95 |
| •Mission Impossible | — | 29.95 | 29.95 |
| •Voodoo Castle | — | 29.95 | 29.95 |
| •The Count | — | 29.95 | 29.95 |
| •Strange Odyssey | — | 29.95 | 29.95 |
| •Mystery Fun House | — | 29.95 | 29.95 |
| •Pyramid of Doom | — | 29.95 | 29.95 |
| •Ghost Town | — | 29.95 | 29.95 |
| •Savage Island I & II | — | 39.95 | 39.95 |
| •Golden Voyage | — | 29.95 | 29.95 |
| •All Star Baseball * | — | — | 19.95 |
| •All Star Bowling * | — | — | 19.95 |
| •Casino Pack * | — | 19.95 | — |
| •Challenge II * | — | — | 16.95 |
| •Chutes and Sharks * | — | — | 19.95 |
| •Dr. Nuttier * | — | 19.95 | — |
| •Gallactic War * | — | 21.95 | 19.95 |
| •Leapfrog * | — | — | 16.95 |
| •Sam Defense * | — | — | 19.95 |
| •Wall Street * | — | — | 19.95 |
| **Others** | | | |
| Demonstration | 69.95 | — | — |
| Speech Editor | 44.95 | — | — |
| Statistics | 44.95 | — | — |
| Extended BASIC | 99.95 | — | — |
| Terminal Emulator II | 49.95 | — | — |
| •Editor Assembler | 99.95 | — | — |
| •Mini Memory | 99.95 | — | — |
| •Word Processing * | — | 99.95 | — |
| Programming Aids I | — | 14.95 | 9.95 |
| Programming Aids II | — | 24.95 | — |
| Math Routine Library | — | 29.95 | 24.95 |
| Electrical Engineering Library | — | 29.95 | 24.95 |
| Programming Aids III | — | 19.95 | — |
| Graphing Package | — | 19.95 | 14.95 |
| Structural Engineering Library | — | 29.95 | 24.95 |

*† Developed by Scott Foresman* *†† Developed by Milton Bradley* ** Developed by Ehninger Associates*

## Expand your TI-99/4 system with these peripheral and printware selections.

| Peripherals/Accessories | Price | Free Software Value Up To: |
|---|---|---|
| TI-99/4A Console | $525.00 | $155.00 |
| Disk Memory Drive | 499.95 | 150.00 |
| Disk Drive Controller | 299.95 | 90.00 |
| Solid State Printer | 399.95 | 120.00 |
| Memory Expansion (32K RAM) | 399.95 | 120.00 |
| P-Code Peripheral | 399.95 | 120.00 |
| RS-232 Accessories Interface | 224.95 | 65.00 |
| Telephone Coupler (Modem) | 224.95 | 65.00 |
| Solid State Speech Synthesizer | 149.95 | 45.00 |
| RF Modulator TV Adapter | 49.95 | 15.00 |

| Peripherals/Accessories | Price | Free Software Value Up To: |
|---|---|---|
| Wired Remote Controllers | 34.95 | These can be combined with other items to qualify for free software |
| Dual Cassette Cable | 14.95 | |
| Monitor Cable | 19.95 | |
| Audio Adapter (Headphone Jack) | 19.95 | |
| Thermal Paper (2-Pack) | 9.95 | |
| **Printware** | | |
| Creative Programming Computer Competency Series—Volume I | 9.95 | |
| Creative Programming Computer Competency Series—Volume II | 9.95 | |
| Creative Programming Computer Competency Series—Volume III | 9.95 | |
| Creative Programming Computer Competency Series—Allstar Projects | 9.95 | |

**Special Limited Time Bonus . FREE Munch Man Command Module!**

Munch Man is TI's version of the tremendously popular Pac Man arcade game. And when you buy any four command modules, you can get MUNCH MAN FREE — in addition to the free software from Unisource.

This is a limited time, TI offer. You must order the four command modules of your choice before May 15, 1982. We will send you proof-of-purchase and a coupon which you can send to TI for your free Munch Man. It's as simple as that. And remember, that's in addition to the free software you get from Unisource in the specials described on the first page of this ad!

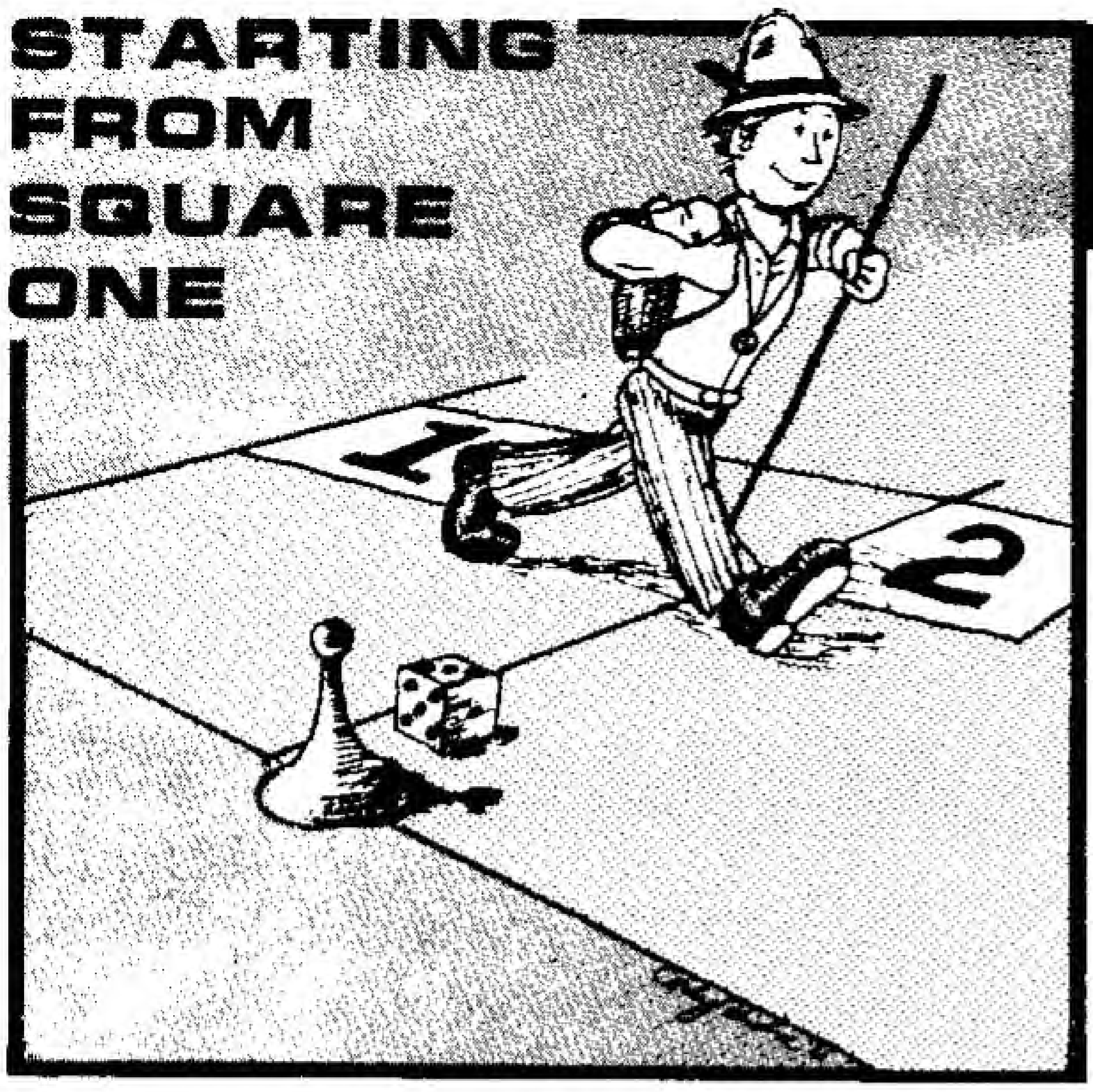
Order by phone - TOLL FREE
**1-800-858-4580**
In Texas call 1-806-745-8835
Phone lines open 8 a.m. - 6 p.m. CST.

Or order by mail
**Unisource Electronics, Inc.**
**P.O. Box 64240**
**Lubbock, Texas 79464**

Orders can be charged to MasterCharge or Visa. $2.00 shipping and handling on software.

**Unisource Electronics, Inc.**

STARTING FROM SQUARE ONE



## Game Designing & Programming FOR BEGINNERS:

**PART 1-** Chuck-A-Luck

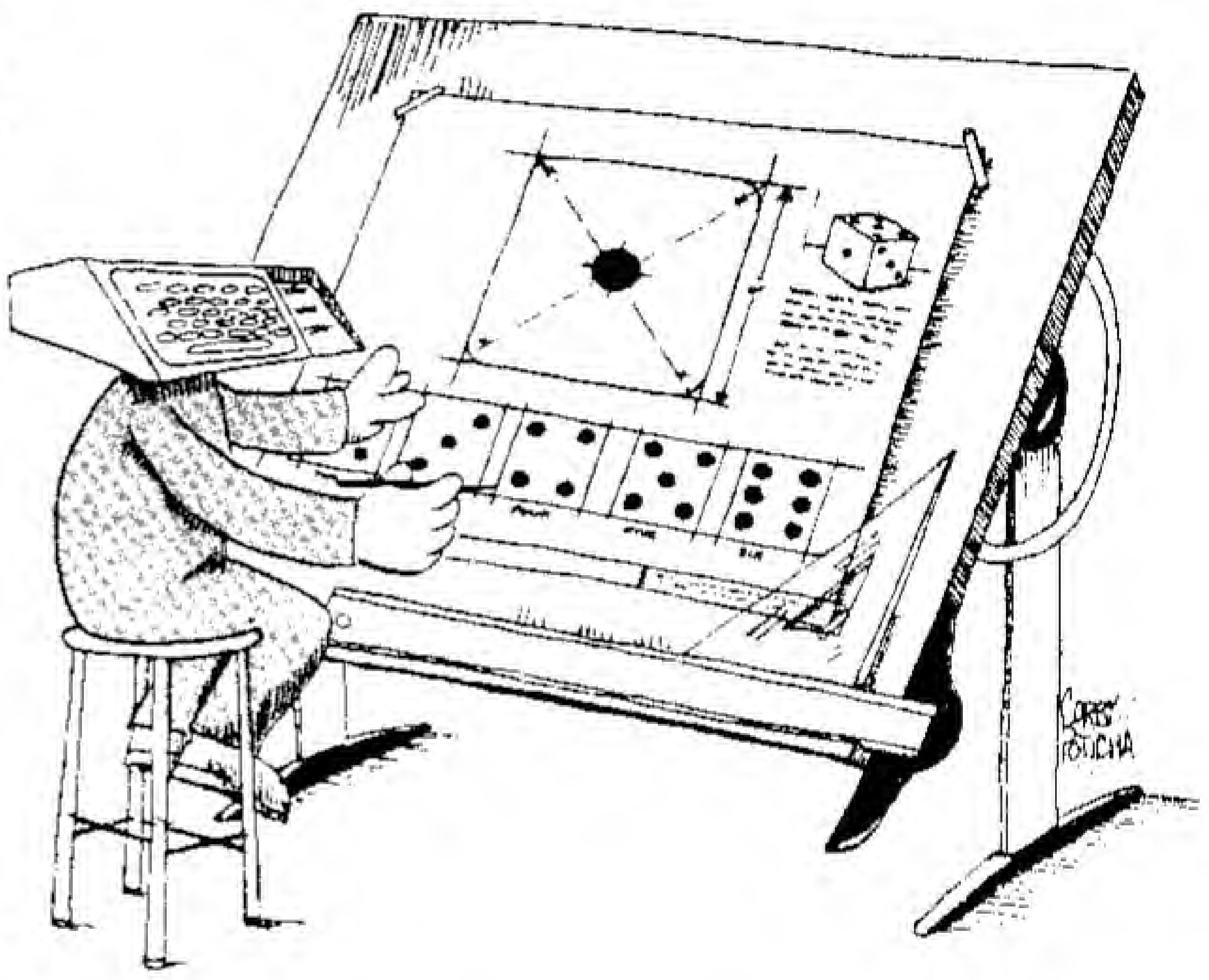
By Samuel D. Pincus

**PART 2-** Fun and Games

By Regena

# CHUCK-A-LUCK

By Samuel D. Pincus



**Have you ever LISTed a program that you bought, and after looking at the listing thought, "that's not so hard . . ."?**

Actually, you're quite correct in *believing* that writing a good program is not really so difficult. But when beginning programmers sit down to translate this belief into a finished program, many wind up confused and frustrated. This can most often be attributed to their accompanying belief that writing a program is just a matter of sitting down at the keyboard and banging away at it -a procedure that is destined to fail.

To explain an alternate approach—one that all experienced programmers use—I'll list the sequence of events that I go through whenever I want to write a program.

First, I sit down and decide what the program is going to do. If it's a game, I write down all the rules (even if I'm making the game up). If it's a business-oriented program, I decide what features it has to have i.e., sorting, saving data, printer output, etc. Without this initial planning, I wouldn't have a goal in mind when I reached subsequent stages.

Second, I design the program. A design is a plan showing the functions (the "whats") that a program contains. For example, a program that plays the game of *Chuck-A-Luck* would contain the following functions: (1) explaining the rules, (2) rolling dice, (3) accepting bets, (4) paying off (or collecting) money, and (5) checking for the final win/loss condition. (See Figure 1 below for the rules of the *Chuck-A-Luck* game that I'll be using as an example). I don't figure out *how* I'll do these things at this time; I just figure out *what* the program has to do.

Third, I group together any "whats" that I feel are different parts of the same top-level module or function.

Figure 1. **CHUCK-A-LUCK Rules**

1. Each player starts with $500.
2. Each player bets an amount of money from $10 to $50 on a dice value from one to six.
3. Three dice are rolled.
4. If no die has a value equal to the value selected by a player, he looses his bet.
5. If one die has a value equal to the value selected by a player, that player receives an amount equal to the amount that he bet.
6. If two dice have that value, the player receives twice the amount he bet.
7. If all three dice have that value, the player receives three times the amount he bet.
8. A player who goes bankrupt is out of the game.
9. The game ends when only one player remains. The remaining player is the winner.
10. If all remaining players go bankrupt at the same time, there is no winner.

For example, giving the rules, generating the dice characters, and getting player names are all part of initialization; so at first, I put them together under the top-level function name of START-UP. Now, I'll write these functions down in a list. For this simple game of *Chuck-A-Luck*, my list of top-level modules looks like this: START-UP, DICE-ROLLS, and END-GAME.

Next, I look closely at each function (or module) and list everything I need to do in each of these modules. For example, the START-UP function will also have to include things like DIMensioning data, asking if rules are needed, asking the number of players, and initializing data fields. The DICE-ROLLS module will have to take bets for each roll, roll the dice (and display them), decide the winners and losers, and recompute new cash balances for each player. The END-GAME routine will have to print an appropriate message after all players go broke or a winner was determined, ask if any player wants to try again, and restart the game.

HCHAR. It would be called by other routines in your program (using the GOSUB statement) whenever they wanted a message displayed on the screen.

But when do we stop *designing* and actually start *writing* the program? There's a different answer for each program and programmer. The idea here is to stop at a point where you feel that you can picture in your mind what the code should look like. For advanced programmers, this may mean that there are fewer modules in a design, and each module will have a lot of lines of code in it. For beginners, I would recommend stopping when each module is self-explanatory to you usually requiring about 10-20 lines of BASIC code.

As I am doing my design, I keep track of the modules by graphically drawing a structured design chart which shows what lower-level modules belong to (are to be part of) each top-level module. After going over all of my top-level modules, my structured design looks like Figure 2, below.

names. But such a program can become user-friendly just by the judicious use of self-explanatory prompts and error messages. Of course, being user-friendly makes for longer and larger programs. I personally don't worry about how much extra memory it requires at first. After all, I can always remove those wonderful messages and replace them with a "NOT POSSIBLE" message if I have to!

By the way, if you stop to think about it, the TI BASIC and EXTENDED BASIC that you work with is very user-friendly. It does things like prompt you for cassette tape I/O and gives you meaningful error messages when you are in EDIT or COMMAND mode.

The second rule that I always follow is that any data that is input into a program must be fully edited—i.e., it must be checked to make sure it is the proper type and in the proper form "expected" by the program. Always! Always! Always!! I said it three times because it is one of the major differences between a professional program (which can be used by anyone without "blowing up'

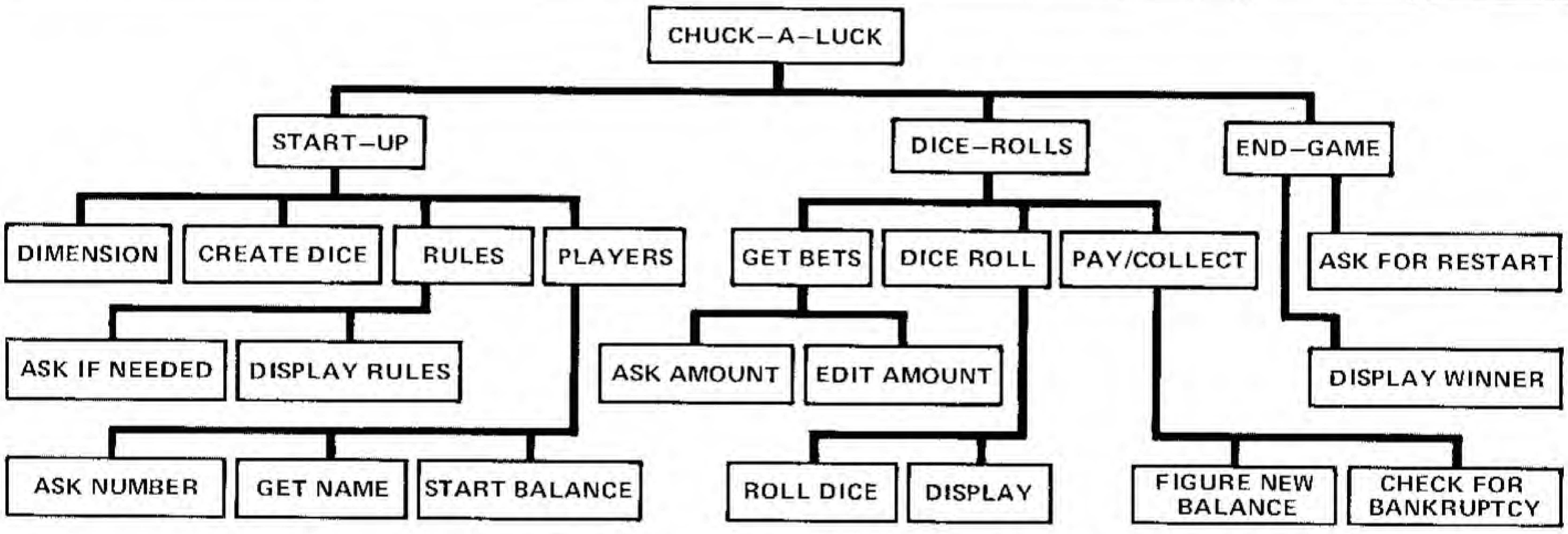


Figure 2. **Structured design for CHUCK–A–LUCK**

Notice that all I have done so far was to write down the "whats" of the program. I haven't looked at the "hows" yet. The technique I have been using is called "top-down design" and consists of breaking a problem or program into its component modules. These new module are themselves broken down into smaller modules. These smaller modules are themselves broken down into even smaller ones until you finally arrive at reasonably sized, easily codable low-level modules.

Sometimes, as you break modules into smaller and smaller routines, you may find at the lowest levels that some modules are duplicated. That means that the same module can belong to (or be used by) more than one higher level module. This kind of routine is called a "subroutine." A good example of a subroutine that you would code in TI BASIC would be a routine to display messages on the screen using CALL

Take a look at the START-UP module. It includes a low-level function called GET NAME. Now look at DICE-ROLLS. It includes a low-level module called EDIT AMOUNT. These routines demonstrate two rules I always follow when I design programs: First, whenever possible, I try to make the program "user-friendly." This includes things like displaying understandable error messages (instead of a cryptic "not possible"), using player names (instead of numbers), and giving prompts that explain what action is required. Too many people write programs that call you PLAYER #1 and tell you to do something by saying things like "CODE?". It only takes a little longer to write a program that says "OK, MIKE, HOW MUCH WILL YOU BET THIS TIME?" And the results are well worth the effort.

Of course, in a business-oriented program, you don't usually ask for people's

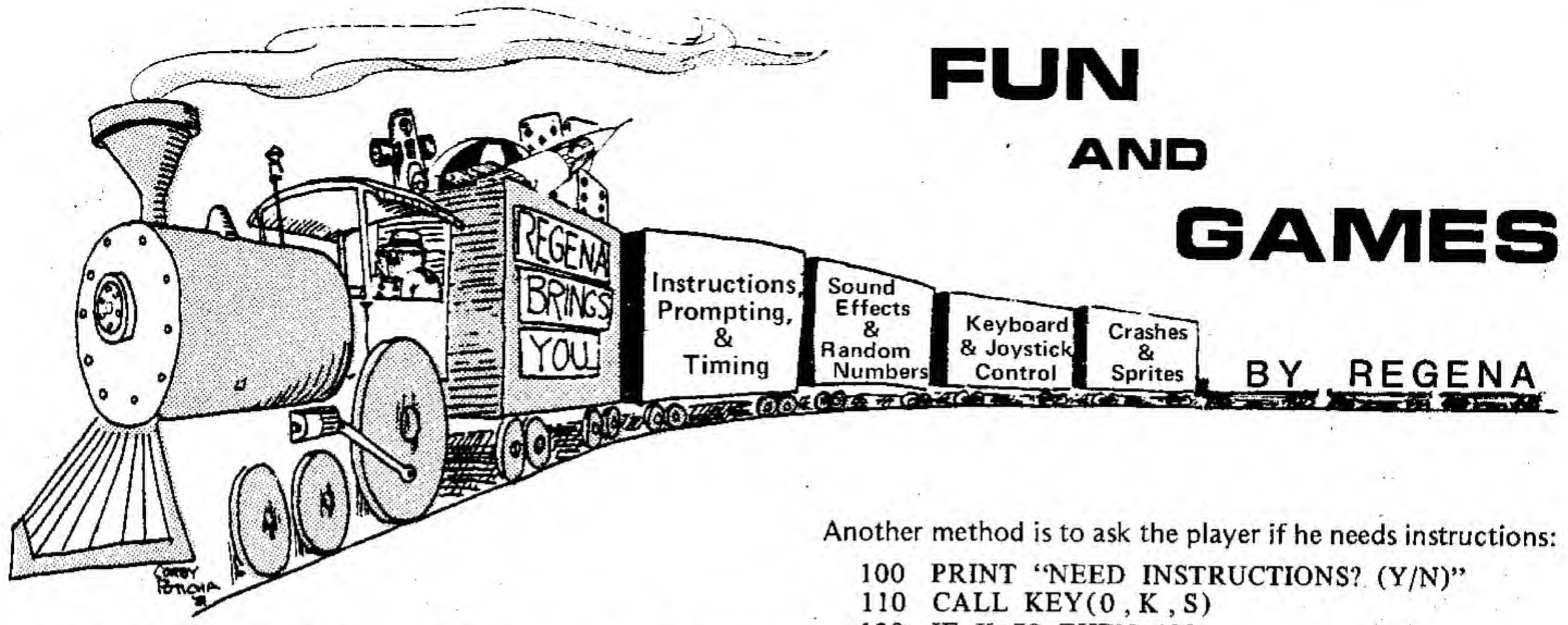
—especially when encountering some strange data input from an unfamiliar user) and a program which is usable only by the person who wrote it.

Some of the rules that I always like to follow include:

1. Making sure that numeric data really is numeric (of course this is something that TI BASIC does for you automatically).
2. Making sure that integers really are integers ( and not decimals or scientific notation).
3. Making sure that the data itself is realistic (always test for maximums and minimums—e.g., making sure nobody bets more money than he has!).

I'm now finished with my design as far as *what* modules are needed. The fifth step in creating a program is to decide what information I need to communicate between these modules. The

Continued on p. 62


REGENA
BRINGS
YOU
Instructions, Prompting, & Timing
Sound Effects & Random Numbers
Keyboard & Joystick Control
Crashes & Sprites

# FUN AND GAMES

BY REGENA

Pssst ! I've got a little secret for you, gang: *Designing* and *programming* your own game can be just as much fun as *playing* games produced by others. And best of all, it's really not as hard as you might think . . .

## Pick an Idea

You can have a maze, a game using dice, a card game, a memory-type game, a board game, a popular sport, a game involving logic, a game using skills or reaction time, some form of hide-and-seek, an adventure, or a myriad of space and shooting games. Still don't have a game plan? Walk through a video arcade to get some ideas.

## Use the Computer

Now this sounds silly, doesn't it, since we're talking about writing computer games. Let me explain. If you write a game of tic-tac-toe or Othello for two players, you're really only utilizing graphics—the game could just as well be played on paper or on the board. But, if you write the game for one person against the computer, then you *are* using the computer to help go through a logic process. Another use of the computer is doing anything with random numbers.

## Write Your Program

Of course, you may just sit at the console and begin programming your game and hope you can remember all the logic. Some programmers like to draw a flowchart. On logic games you may like "tree diagrams"—i.e., if the player does one option you branch one way; then depending on the next choice, you branch again, and so forth. Other programmers prefer a structured approach—each process of the game is in a subroutine and the main program calls the subroutines in order. This type of program is easy to evaluate and easier for other programmers to follow than a program that has GOTO statements all over the place.

## Include Instructions

Many players are anxious to play the game and won't read anything that comes with the game program, so it is wise to include simple instructions within your program. Players who are playing the game a second time, however, won't want instructions, so you must try to satisfy everyone. One method is to print the instructions on one screen with "PRESS ANY KEY TO START" at the bottom of the screen. The player can then look at the screen as long as he wants or he may immediately press any key to start the game.

```
100  PRINT "PRESS ARROW KEYS TO GO"
110  PRINT "LEFT OR RIGHT."
120  PRINT "PRESS 'F' TO SHOOT."
130  PRINT : : : "PRESS ANY KEY TO START."
140  CALL KEY(0 , K , S)
150  IF S<1 THEN 140
160  Program continues for game
```

Another method is to ask the player if he needs instructions:

```
100  PRINT "NEED INSTRUCTIONS? (Y/N)"
110  CALL KEY(0 , K , S)
120  IF K=78 THEN 400
130  IF K<>89 THEN 110
140  Program prints instructions.
400  Program continues for game.
```

If the player presses "Y", instructions will be printed; if he presses "N", the game starts. Any other key pressed is ignored.

Be sure the instructions are as clear and concise as possible. Use enough blank lines to make the instructions easy to read. Make sure words are not divided at the ends of lines, be sure to spell correctly, and use correct grammar.

## "Dummy-Proof" Your Game

A nicer way of saying this is to make your program "user-friendly." This means consider all possibilities of input. You never know what some other player will try to do. If you have to answer "yes" or "no", can the player just press Y or N, or does he need to spell out and ENTER the answer? Pressing one key has a lot less chance of error than using INPUT. What if you ask for a *number*, and a *letter* is pressed? What if you ask for a choice of 1 through 4 and the number 7 is pressed? If the player needs to use the arrow keys, is there a default value if he hits another key, or is that key ignored—or worse yet, does the program crash?

## Check for Speed and Captivation

You don't want the player to fall asleep in between moves. If you have moving objects in your game, the player wants them to be as fast as possible. The main hints here are to have the moving object be just one character and to minimize the logic between moves. Also, the more objects you have to move, the longer it will take. Another hint is that if you don't need to worry about scrolling (lines moving up the screen), PRINTing characters is faster than CALL HCHAR or VCHAR.

## Look Through Your Listing

If you use the same group of lines several times, use a GOSUB, and place the subroutine near the beginning of the program. For example, a subroutine to print a message M$ on Row X starting in Column 1 is:

```
180  FOR J=1 TO LEN(M$)
190  CALL HCHAR(X , J , ASC(SEG$(M$(J , 1)))
200  NEXT J
210  RETURN
```

Within the program, the message and row number are defined then the subroutine called:

```
1740 M$="BLUE WINS THIS TIME."
1750 X=22
1760 GOSUB 180
```

Check for unnecessary statements. I have seen a few listings that contain some coding that can never be executed, or a subroutine that is never called. Other cases may occur because of editing. For example:

```
900  GOTO 920
910  X=25
920  GOTO 980
     OR
900  GOTO 910
910  Z=Z+1
     OR
900  IF X=A THEN 700 ELSE 910
910  GOTO 980
```

## Test Your Game

Again, check all possibilities. If you say your spaceship can move to the right and to the left, be sure to check *both* directions. Make sure positive and negative numbers work correctly in your calculations (you may want to use the ABSolute value). Check the scoring to see if it is adding correctly. Test the possibility of hitting the wrong key. Test moving objects at the edges of the screen.

## Specific Game Coding

### Random Numbers

Be sure to use the statement RANDOMIZE before using RND so each game played will be different. If random numbers are computed several different places, consider using RANDOMIZE before each RND to ensure total randomization throughout the game. Sometimes a single RANDOMIZE statement at the beginning of the program does not work.

Shooting dice would need a random number from 1 to 6:

```
100  RANDOMIZE
110  D1=INT(RND*6)+1
```

In a space program or skill-type game you may want to place obstacles at random positions. If you have several objects, DEFine a few functions at the beginning of the program, then they can be used more easily in the coding later:

```
100  DEF RX=INT(RND*24)+1
110  DEF RY=INT(RND*29)+2
120  CALL CLEAR
130  RANDOMIZE
140  FOR I=1 TO 5
150  CALL HCHAR(RX,RY,65)
160  NEXT I
170  CALL VCHAR(RX,RY,66)
180  STOP
```

The DEFinition statements must be numbered lower than the statements in which the functions are used. Lines 140-170 place five As and one B in random X and Y positions for X from 1 to 24 and Y from 2 to 30.

Another use of random numbers is choosing a random message or procedure. For example,

```
500  PRINT A$(INT(RND*9)+1)
```

chooses one of nine messages previously stored in the A$ array. For random subroutines, the coding would be

```
510  ON INT(RND*5)+1 GOSUB 200,250,300,350,400
```

Games using a deck of cards may use an array to keep track of which cards are dealt. You may use C$ (52) for the 52 cards, or a two-dimensional array C(13,4) where the first parameter is the number chosen and the second is the suit. An example for choosing ten cards follows. The values in the card array are initially zero. As a card is chosen, the corresponding C element is set equal to 1. In the following example I printed the card values, but remember you really should use the TI graphics to *draw* the cards.

```
100 REM  CARDS
110 CALL CLEAR
120 DIM C(13,4),A$(13)
130 DATA ACE,2,3,4,5,6,7,8,9,10,JACK,QUEEN,KING
140 FOR J=1 TO 13
150 READ A$(J)
160 NEXT J
170 SUIT$(1)="HEARTS"
180 SUIT$(2)="CLUBS"
190 SUIT$(3)="DIAMONDS"
200 SUIT$(4)="SPADES"
210 PRINT "TEN CARDS CHOSEN:"::
220 RANDOMIZE
230 FOR I=1 TO 10
240 N=INT(13*RND)+1
250 S=INT(4*RND)+1
260 IF C(N,S)=1 THEN 240
270 PRINT A$(N);" OF ";SUIT$(S)
280 C(N,S)=1
290 NEXT I
300 STOP
```

One more use of RND is for choosing random sounds. The CALL SOUND statement requires a frequency between 110 and 44733. Of course, most people cannot hear frequencies above 15,000; however, your dog may enjoy the higher frequencies. This statement plays a sound frequency between 110 and 2110:

```
300  CALL SOUND(200,INT(RND*2000)+110,0)
```

You may wish to use random sounds while you're placing objects randomly on the screen.

### Sound and Noise

A lot of the fun in programming games is choosing the sound effects to fit your game. The following is a program that demonstrates the "noises" available on the TI-99/4A.

```
100  REM NOISE
110  FOR I=-1 to -8 STEP -1
120  CALL SOUND(4000,I,0)
130  CALL CLEAR
140  CALL SCREEN(ABS(I)+2)
150  PRINT "NOISE NUMBER";I
160  NEXT I
170  GOTO 110
180  STOP
```

Listen to these noises and choose what you need for your game. You can make crashing noises, explosions, airplane or car motors, splats, bounces, rocket boosters, missile fire, or whatever you need. The noises may be varied by adding another set of sound frequencies and loudnesses.

### Time

Since the 99/4A does not have an accessible real-time clock, time may be simulated by placing a counter in the CALL KEY routine or other loop that is executed regularly. The following example shows a counter as you move the asterisk up and down with the up arrow and down arrow (E and X) keys. After a time of 100, the number of moves you made is printed. You will notice that if you press a key the counter moves more slowly than if no key is pressed, so the counter is not as even as a metronome but good enough for games.

```
100 REM  TIMING
110 CALL CLEAR
120 X=12
130 CALL HCHAR(X,15,42)
140 TIME=0
150 CALL KEY(0,K,S)
160 TIME=TIME+1
170 FOR I=1 TO LEN(STR$(TIME))
180 CALL HCHAR(22,I+3,ASC(SEG$(STR$(TIME),I,1)))
190 NEXT I
200 IF TIME=100 THEN 350
210 IF K<>69 THEN 240
220 DX=-1
230 GOTO 260
240 IF K<>88 THEN 150
```

software
NORTON
software

**SOFTWARE FOR THE TI 99/4 COMPUTER**

Fast, well, structured programs which fully exploit the potential of the TI computer for less than you expect to pay. Here is just a short list of what we have:

Interactive graphics
-a fast way to draw high resolution graphics

Speech adder
-expands the TI's speech capacity to include most of the English language

Music synthesizer

High resolution games which test the limits of your skills

**Send for free brochure-**
**Norton Software**
**Box 575**
**Picton, Ontario**
**K0K 2T0**

NEW! **QUALITY SOFTWARE FOR TI 99/4 & 4A** NEW!

**3-D MAZE** - Enter the world of 3-D MAZE. Look down the hallways, explore them, find the dead ends and the loops. Try to remember in what direction you are going and then find the exit! There are hundreds of mazes, some easy, some very difficult. Come, try to conquer the world of 3-D Maze. Ages 12 to Adult. X Basic $14.95.

**BOMB SQUAD** - Time is ticking away. Only 16 seconds left. To stop the clock the timer circuit must be cut. But which one is it? One more test and you should know. The question is, can you figure it out in time?

For high intensity excitement this is the best game we've seen. As the time relentlessly ticks away you must make tests on the bombs circuits and use logic to deduce which wires to cut. Over 20 levels of difficulty make this game a challenge for the beginner or the expert. Ages 12 to Adult. X Basic $14.95.

| | | |
|---|---|---|
| • **METEOR SHOWER** | X Basic | $ 9.95 |
| • **SKI** | Basic | $14.95 |
| • **LAND ON MARS*** | Basic | $12.95 |
| • **LASER SHIELD** | Basic | $14.95 |
| • **SPACE BATTLE 2056** | Basic | $13.95 |
| • **MR. FROG** | X Basic | $14.95 |
| • **BARNYARD FUN** | Basic | $14.95 |

***Dragon option included!**

All programs on cassettes, complete instructions included. Plus $1.50 shipping/handling, Mn. residents add 5% sales tax. To order or for more information:

ASD&D

**AMERICAN SOFTWARE DESIGN & DISTRIBUTION CO.**
**P.O. BOX 46**
**COTTAGE GROVE, MN 55016-0046**

```
250 DX=1
260 CALL HCHAR(X,15,32)
270 X=X+DX
280 IF X>0 THEN 300
290 X=24
300 IF X<25 THEN 320
310 X=1
320 CALL HCHAR(X,15,42)
330 MOVES=MOVES+1
340 GOTO 150
350 PRINT "MOVES =";MOVES
360 STOP
```

Following is another example of a way to time a process—in this case, typing your name.

```
100 REM  SPEED TEST
110 CALL CLEAR
120 TIME=0
130 PRINT "TYPE NAME THEN PRESS ENTER.":::::
140 FOR Y=3 TO 28
150 CALL KEY(0,K,S)
160 TIME=TIME+1
170 IF S<1 THEN 150
180 IF K=13 THEN 210
190 CALL HCHAR(21,Y,K)
200 NEXT Y
210 PRINT "TIME =";TIME
220 STOP
```

An accurate way to delay for a specific length of time in your program is to use CALL SOUND for the number of milliseconds you need. Use 30 for the volume level and a very high frequency if you don't want to hear anything. While the CALL SOUND statement is being executed you may also be doing graphics or calculations. To end your timing device you will need another sound statement with a duration of 1. The following example illustrates how the CALL SOUND statements may be used for a rocket countdown.

```
100  FOR I=10 TO 1 STEP -1
110  CALL SOUND(1000,44000,30)
120  PRINT I
130  NEXT I
140  CALL SOUND(1,44000,30)
150  Program continues for rocket blastoff.
```

## Arrow keys

In games where you move a character up, down, left, or right, you may wish to have the player press the arrow keys. The arrows are on the keys E, D, X, and S. A CALL KEY statement is used to receive the player's input, then the program branches depending on which arrow is pressed. Any other key pressed should be ignored so your program doesn't crash with bad values.

The following routine will draw a trail of asterisks as you press the arrow keys. Remember, you must consider the edges of the screen or you will get a "BAD VALUE" message. Lines 270-340 test for the edge values and will keep the asterisk at the edge position.

```
100 REM  MAKE-A-TRAIL
110 CALL CLEAR
120 X=12
130 Y=15
140 CALL HCHAR(12,15,42)
150 CALL KEY(0,K,S)
160 IF K<>69 THEN 190
170 X=X-1
180 GOTO 270
190 IF K<>68 THEN 220
200 Y=Y+1
210 GOTO 270
220 IF K<>88 THEN 250
230 X=X+1
240 GOTO 270
250 IF K<>83 THEN 150
260 Y=Y-1
270 IF X>=1 THEN 290
280 X=1
290 IF X<=24 THEN 310
300 X=24
310 IF Y>=1 THEN 330
320 Y=1
330 IF Y<=32 THEN 350
340 Y=32
350 CALL HCHAR(X,Y,42)
360 GOTO 150
370 STOP
```

Remember, there are many ways of coding to get the same result, and the examples presented here are merely that—just *examples.* The following routine illustrates another way to use the arrow keys to move a character. This time the previous character is deleted. Also, Lines 330-410 will make the asterisk scroll to the other side of the screen instead of staying at the edge.

```
100 REM  MOVE-A-STAR
110 CALL CLEAR
120 X=12
130 Y=15
140 CALL HCHAR(X,Y,42)
150 CALL KEY(0,K,S)
160 IF K<>69 THEN 200
170 DX=-1
180 DY=0
190 GOTO 310
200 IF K<>68 THEN 240
210 DX=0
220 DY=1
230 GOTO 310
240 IF K<>88 THEN 280
250 DX=1
260 DY=0
270 GOTO 310
280 IF K<>83 THEN 150
290 DX=0
300 DY=-1
310 CALL HCHAR(X,Y,32)
320 X=X+DX
330 IF X>0 THEN 350
340 X=24
350 IF X<25 THEN 370
360 X=1
370 Y=Y+DY
380 IF Y>0 THEN 400
390 Y=32
400 IF Y<33 THEN 420
410 Y=1
420 CALL HCHAR(X,Y,42)
430 GOTO 150
440 STOP
```

A more compact approach to automatic scrolling is to replace Lines 330-360 and 380-410 with these two lines:

```
330   X=INT(24*((X-1)/24-INT((X-1)/24)))+1
380   Y=INT(32*((Y-1)/32-INT((Y-1)/32)))+1
```

## Split Keyboard

A split keyboard is used when two competing players or teams are interacting with moving objects on the screen. Instead of CALL KEY (0, KEY,STATUS), you will need to receive input with CALL KEY (1,KEY1,STATUS1) and CALL KEY(2,KEY2, STATUS2). You may wish to use a *Video Games 1* Command Module overlay for the arrow keys. You'll notice the arrow keys for the right side of the keyboard are keys I, J, K, and M. The key codes returned in CALL KEY are 5 for up, 2 for left, 3 for right, and 0 for down for both sides of the split keyboard. *Note:* There is a slight problem in testing for zero on the 99/4A console, so use logic such as IF KEY2+1<>1 instead of IF KEY2 <> 0. It is also wise to avoid using SHIFT, ENTER, G, B, slash, semi-colon, comma, period, and the space bar for key input (such as firing a missile) because the key codes for these keys are different on the 99/4 and 99/4A. You will want your game to work on both consoles so you can share with others.

An example of the logic for two players and a split keyboard is shown in Lines 910-1510 in the game "MAZE RACE" in this issue of *99'er Magazine*. Also, both games in Kelley's Korner of the September/October issue utilized the split keyboard approach.

## Joysticks

Enter the sample programs that come with your TI Wired Remote Controllers to get an idea how to program movement with one or two joysticks. Keep in mind that CALL JOYST (KU, X, Y) returns X and Y values of 0 and plus or minus 4 depending on the position of the lever. By the way, don't get these X and Y values confused with the X- and Y-coordinate values for HCHAR or VCHAR.

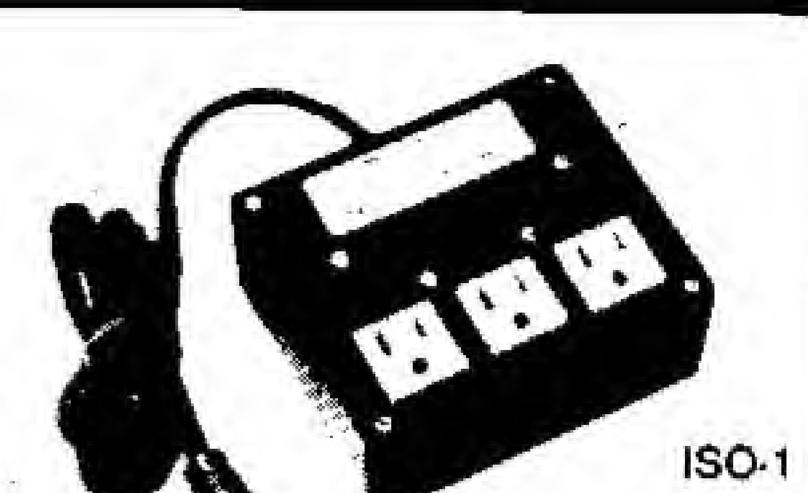
Following is a sample program that allows the player to move the asterisk either with the arrow keys or with a joystick. Line 150 is CALL KEY statement. If no key on the keyboard is pressed, all the arrow key logic is skipped and CALL JOYST (Line 330) is executed; if a key has been pressed, then the joystick logic statement (lines 330-350) are skipped.

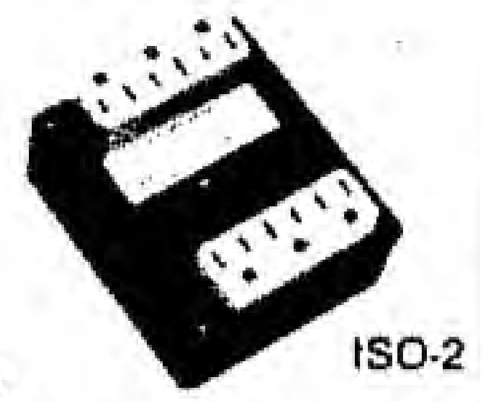
```
100 REM  JOYSTICKS
110 CALL CLEAR
120 X=12
130 Y=15
140 CALL HCHAR(X,Y,42)
150 CALL KEY(0,K,S)
160 IF S=0 THEN 330
170 IF K<>69 THEN 210
180 DX=-1
190 DY=0
200 GOTO 320
210 IF K<>68 THEN 250
220 DX=0
230 DY=1
240 GOTO 320
250 IF K<>88 THEN 290
```

Continued on p. 79

**DISK DRIVE WOES?**
**PRINTER INTERACTION?**
**MEMORY LOSS?**
**ERRATIC OPERATION?**

# Don't Blame The Software!





Power Line Spikes, Surges & Hash could be the culprit! Pat. #4,259,705
Floppies, printers, memory & processor often interact! Our patented ISOLATORS eliminate equipment interaction AND curb damaging Power Line Spikes, Surges and Hash. Guaranteed!

- **ISOLATOR (ISO-1) 3 filter isolated 3-prong sockets; Integral Surge/Spike Suppression; 1875 W Maximum load, 1 KW load any socket ........ $69.95**
- **ISOLATOR (ISO-2) 2 filter isolated 3-prong socket banks; (6 sockets total); Integral Spike/Surge Suppression; 1875 W Max load, 1 KW either bank ........ $69.95**
- **SUPER ISOLATOR (ISO-3) similar to ISO-1 except double isolation & Suppression ........ $104.95**
- **SUPER ISOLATOR (ISO-11) similar to ISO-2 except double isolation & Suppression ........ $104.95**
- **MAGNUM ISOLATOR (ISO-17) 4 Quad Isolated sockets; For ULTRA-SENSITIVE Systems ........ $181.95**
- **CIRCUIT BREAKER, any model (Add-CB) ........ Add $9.00**
- **CKT BRKR/SWITCH/PILOT (-CBS) ........ Add $17.00**

**AT YOUR DEALERS**

**MasterCard, Visa, American Express**
**ORDER TOLL FREE 1-800-225-4876**
**(except AK, HI, PR & Canada)**

*Electronic Specialists, Inc.*
*171 South Main Street, Natick, Mass. 01760*
Technical & Non-800: 1-617-655-1532

**FREE** CATALOG & NEWSLETTERS

for TI-99/4A USERS

- Catalog includes hundreds of programs of all types : Education & Science, Business and Professional applications, Games, Music, Utilities, Languages, and Demos. Largest collection of educational software for the TI-99/4(A) ever assembled.
- TI SOURCE — For latest developments and reviews of new software.
- TI LOGO SOURCE — Keeps TI LOGO users abreast of latest developments, and reviews LOGO procedures.

*Write or Phone*
*For Your Free Catalog & Newsletters Today :*

MICROCOMPUTERS CORPORATION
34 Maple Ave.
Armonk, NY 10504
*Tel. (914) 273-6480*

## Pewterware

*Known For Its Outstanding Graphics, Makes This* Special Offer *For 99'er Readers:*

- Bluegrass Sweepstakes
- Match Wits
- Decathlon
- Challenge Poker

| | |
|---|---|
| One Game Program .. | $ 9.00 PPD |
| Any Two ............ | 15.00 PPD |
| Any Three ........... | 18.00 PPD |
| All Four Programs ..... | 20.00 PPD |

*Programs are on cassette and as always,*
Satisfaction is Guaranteed

Send Check or Money Order To :

Tom Krohn
834 Christy Court
Pensacola, FL 32508

## Computer-Ed.

Of
Carmel, NY & Los Angeles, CA

## 99/4(A) Programs

- K-6 Reading
- K-6 Language Arts
- K-6 Math
  - Follow-up Games and Worksheets
  - Classroom Tested
  - Also Available for TRS-80 Level II

*For*
**FREE** Catalogue
*Write To:*

Computer-Ed.
1 Everett Rd.
Carmel, NY 10512

****

## TI-Asteroids

- 99/4 version of Popular Arcade Game
- Finest Extended BASIC Game Available
- Rated **Four Stars** By National Users Group
- Four Skill Levels
- "High Score To Date" Feature in Both Versions

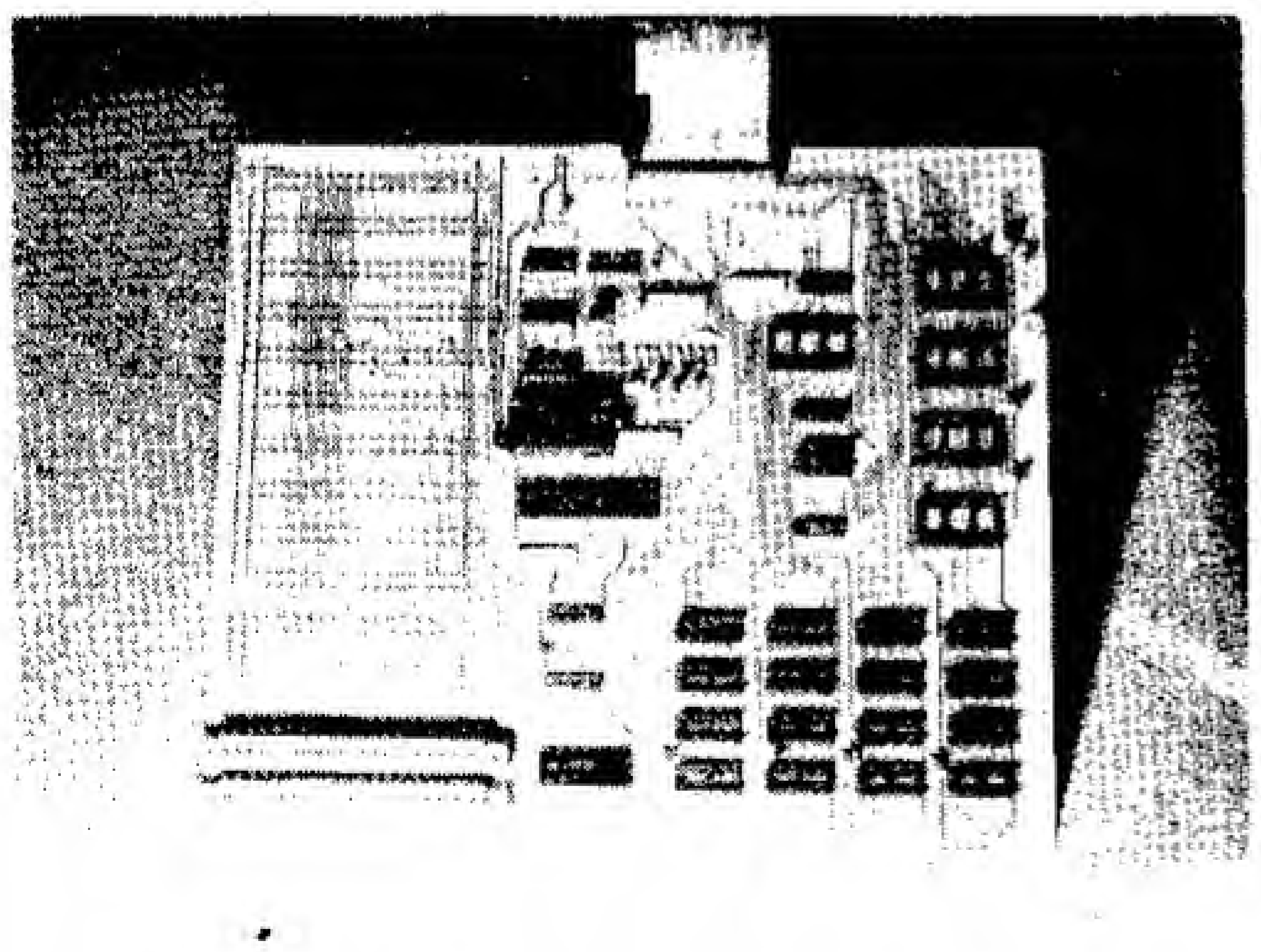
$17.50
for Cassette or Disk Version

*(NJ residents add 5% tax)*

To order or for information write :

FFF SOFTWARE
P. O. Box 4169
Trenton, NJ 08610

## FOR TI UNIVERSITY BOARD OWNERS:



MEMORY AND I/O EXPANSION MODULE (MEM)

- 8K MEMORY EXPANSION--RAM/EPROM MIX
- I/O EXPANSION TO 480 SINGLE-BIT PORTS
- BUILT-IN EPROM PROGRAMMER

EXPAND YOUR UNIVERSITY BOARD!

INCREASE THE VERSATILITY OF YOUR UNIVERSITY BOARD (TM 990/189) BY KEEPING YOUR UTILITY PROGRAMS "ON LINE". THE MEMORY AND I/O EXPANSION MODULE MAKES THE UNUSED PORTION OF THE UNIVERSITY BOARD'S MEMORY SPACE AVAILABLE WITH CONVENIENT PARTITIONING. YOU CAN HAVE UP TO 8K OF ADDITIONAL MEMORY ON LINE, WITH ANY MIX OF READ/WRITE MEMORY AND EPROM; JUMPER SELECTABLE ON 1K BOUNDARIES WITH POWER ON.

FOR I/O-INTENSIVE APPLICATIONS, THE MEM HAS A 15-SLOT I/O SECTION WITH EACH SLOT ADDRESSING 32 BITS ON THE UNIVERSITY BOARD'S UNIQUE CRU BUS. THIS GIVES SINGLE- OR MULTI-BIT ACCESS TO 480 INDIVIDUAL I/O POINTS.

THE MEM HAS A BUILT-IN PROGRAMMER FOR TMS 2708 AND 2716 EPROMS, WITH PROGRAMMING SOFTWARE CAPABLE OF BEING LOCATED AT FOUR ADDRESSES. THIS SOFTWARE ALLOWS YOU TO COPY NEW CODE FROM EXISTING ROMs OR DIRECTLY FROM THE READ/WRITE MEMORY USED TO DEVELOP THE CODE.

MEMORY AND I/O EXPANSION MODULE
(Assembled and tested with complete User's Manual)
$199.95 plus $5 postage and handling
(Texas residents please add 5% sales tax)

**George Goode & Associates, Inc.**

12606 GREENVILLE AVENUE
SUITE 100
DALLAS, TX 75243
PHONE (214) 690-0595

# How To Write A BASIC Program That Writes BASIC Programs

## PART 2: THE RULES OF MERGE FORMAT

By John Clulow

In the previous issue, MERGE format was discussed in connection with TI's *Programming Aids III.* When an Extended BASIC program is saved with the MERGE option (disk only), a data file is written such that each record in the file contains a coded representation of one line of BASIC code. This file can then be loaded into program memory with the MERGE command. Because the file is a data file, it can also be generated by a BASIC program. If all of the rules of MERGE format are observed, the file is indistinguishable from one created by saving a program with the MERGE option and can be loaded into program memory with the MERGE command. Thus an Extended BASIC program can, in effect, write another Extended BASIC program.

One can think of a variety of contexts in which this program generation capability could be used. For instance, a program might allow preparation of music or graphics in an interactive, "high level" format and then use this data to write a BASIC program or subroutine which produces the music or graphics display.

### File Structure

The MERGE format file consists of sequentially organized records each corresponding to one line of BASIC code. Records are of variable length with a maximum length of 163 display format characters. The OPEN statement for a MERGE format file might be:

```
OPEN #1:"DSK1.FILENAME",VARIABLE 163
```

### Record Structure

Records in the file each represent a line of BASIC as strings of ASCII characters. The ASCII codes of the first two characters represent the line number, the last character designates "end-of-line", and the BASIC statement(s) are represented in coded form in between.

Let's consider first how line numbers are represented. You are probably aware that code numbers are associated with the character patterns used to display information. The character associated with a code can be obtained with the CHR$ function; PRINT CHR$(65) displays the pattern of the character with ASCII code 65 on the screen—the letter A. (ASCII, by the way, stands for American Standard Code for Information Interchange.)

While some ASCII characters, like the letter A, have an associated pattern, others do not. However, any of the 256 ASCII characters can be accessed with the CHR$ function and subsequently used in strings just like any of the more familiar characters. PRINT CHR$(32)&CHR$(255) displays two characters. Neither has a pattern so neither can be seen, but the computer is able to recognize each character nevertheless.

A character consists of a "byte," and a byte can be thought of as an eight-place binary number. Just as the decimal number system contains 10 digits (0-9) the binary system contains two digits, 0 and 1. In the decimal system, the first place to the left of the decimal point counts in units of one. Each successive place counts in units of the number base 10, multiplied times the units of the preceeding place—i.e., 1's, 10's, 100's, 1000's, etc. Similarly, the first place in a binary number counts 1 and successive places in units of the number base 2, multiplied times the units of the preceeding base; i.e., 1's, 2's, 4's, 8's, 16's, etc. Thus the eight-place binary number 00110001 is equivalent to 0+0+32+16+0+0+0+1 or 49 in decimal. The binary number 11111111 is equivalent to 255 (128+64+32+16+8+4+2+1), and this is the largest ASCII code because it is the largest number that can be represented with a byte. The 256 ASCII characters are thus numbered from 0 through 255.

The decimal equivalent of the ASCII code is used to represent the line number; but with only one byte, the largest line number which could be represented is 255. To allow representation of higher line numbers, a second character is added giving a total of 16 binary places. Applying the same principle used above, the places count (from right to left) in units of 256 (128*2), 512, 1024, etc. When placed in the first two positions of a MERGE format record, CHR$(2)&CHR$(8) —i.e., 00000010 00001000 — would represent the line number 520 (512+8). A quick method of determining the decimal representation of any two characters is to multiply the code of the first by 256 and add the code of the second. In the above example, 520 = 2*256+8.

The highest allowable line number in TI BASIC is 32767 (01111111 11111111 or CHR$(127)&CHR$(255)). Adding the left digit gives the end-of-file mark used in MERGE format, equivalent to a line number of 65535. These two bytes, CHR$(255)&CHR$(255) must be in the first two positions of the last record in a MERGE format file.

Just as these two characters signal the end of the file, the byte CHR$(0) is used to signal the end of each line. This character must be the last one in each record.

### MERGE format Code

This brings us to the question of what to put between the line number and end-of-line mark and before the end-of-file mark; viz., the coded BASIC statements. Many elements which comprise Extended BASIC statements are listed in

Continued on p. 54

Sunshine Software

• • CRIBBAGE • •
Play against the TI-99/4. Computer plays the able opponent.

• • QUBIQ • •
A unique mind-boggling puzzle. Full Color & Graphics.

• • SPIRAL-GRAPHICS • •
Creates thousands of amazingly beautiful patterns in High Resolution. You specify the pattern, the computer does the rest.

• • OBSTACLE COURSE • •
Can you make it through this game of logic? Deduction tells you where the obstacles are.

*$8.95 each; 4 for $24.95*
*Cassette only; TI BASIC*
*Complete documentation included*

Sunshine Software
520 Milton Lane
Hoffman Estates, IL 60194
Tel (312)-885-2862

**The Cube**
3 x 3 Puzzle
**Quadcube**™
4 x 4 Puzzle

Vivid color 3-D graphic simulations with scramble and unscramble commands
Cassette-TI99/4-4A

$14.95 each · postpaid
Linear Aesthetic Systems
POB 23 · West Cornwall
Connecticut 06796

**TI-COUNT**
**Small Business System**

| | |
|---|---|
| • **General Ledger** (professionally approved) | 650 |
| • **Accounts Payable** (writes checks) | 300 |
| • **Accounts Receivable** (prepares bills) | 300 |
| • **Inventory** (1400 items per disk) | 300 |
| • **Mailing list** (renewal letters) | 650 |

**Mini Computer Performance At Micro Prices**

**Only For The 99/4**

**For information on dealer program, write or call.**

**To order, send check to:**
**PIKE CREEK COMPUTER CO., INC.**
2 Galaxy Drive
Newark, Delaware 19711
(302) 239-5113

# PROGRAMMING HINTS ........ BY REGENA

## WHO IS REGENA???

Twice each month, like clockwork, a peculiar event occurs. An air freight delivery truck rolls up to our door and drops off a mysterious parcel. Each time, the outside wrapper carries markings from a different country. One thing, however, always remains the same—the two-word return address: "From Regena."

The contents of each package are similar—manuscripts and tapes of articles and programs for and about the TI-99/4 computer, plus a one-line note: "Hope you can use this—Regena." We've never been able to contact this mysterious programmer. Since we presume that he or she gets to see the magazine, it would seem logical that Regena must be a subscriber. But we can't be certain, as large numbers of issues also get sent in bulk to dealers and distributors all over the U. S. and abroad.

Also, there's the matter of the formal letter we received one day from a Zurich bank, with instructions to mail Regena's payment checks to a numbered Swiss account. All very secret and strange indeed . . .

Here at the magazine, we've tried to figure it out: Why would anyone who goes to all this trouble to keep their identity and whereabouts such a secret be writing for a computer magazine? Unless, of course . . . But no, that's the stuff that spy thrillers and James Bond movies are made of . . .

Regena's manuscripts (each typed on a different machine, on paper bearing different watermarks) and program tapes (long strips of reel-to-reel recording tape, rolled up tightly and inserted into 35 mm film cans) have been thoroughly analyzed for clues. It is true, we have found a few, but we suspect, however, that they were deliberate "red herrings." Perhaps you, our readers, have detected some clues in the many articles and programs of Regena that we've published. Or perhaps one of you has actually made contact with our mysterious programmer. In any case, please send us your ideas or information.

And Regena—if you're reading these words—won't you please "come in from the cold . . . "

## #1: Testing For Equality

In the statement

```
100 IF A=B THEN 200
```

the test for equality will fail if A and B do not match digit for digit to the full precision of the TI-99/4 computer (13 decimal places). In most programming situations the tolerance level is not that critical, and you may allow accuracy to only two or three decimal places. The equivalent statement that will test for equality out to three decimal places is:

```
100 IF ABS(A–B)<0.001 THEN 200
```

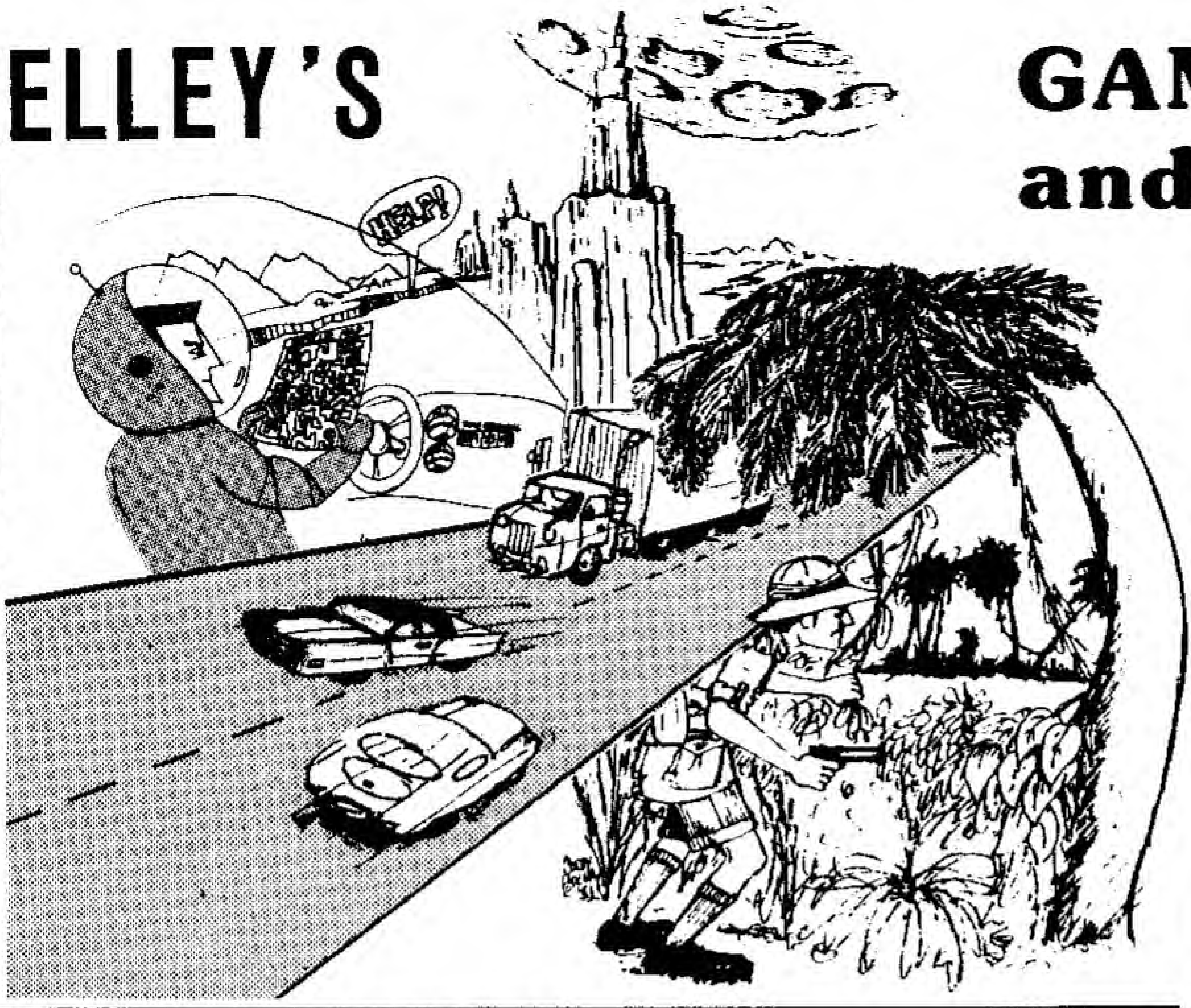
## #2: Q Can Get You! (If You Compose Music on the Original TI–99/4 Keyboard)

Do not use "Q" for quarter notes in writing music programs. It is too easy to SHIFT Q and lose your program especially since Q is between a parenthesis and a comma (both of which require the SHIFT key) in the CALL SOUND statement. Perhaps "T" for "time" or "M" for "metronome setting" or even "N" for "note" could be used.

The use of "Q" in any variable name should be avoided in all programs.

# KELLEY'S KORNER

# GAMES of TIME and DIRECTION



Look closely, 99'ers
And you shall see,
A race against time,
Not one, but three . . .

First, it's a maze,
Where you and a foe
Race to the finish,
And lose if too slow.

Then, it's a rescue
On Venus, Mars, and the moon,
Where strategy and skill
Are surely a boon . . .

Finally, comes a freeway,
As rush hour draws near,
To dodge the traffic,
You must steer clear and veer.

So go for it, 99'ers,
With games ready to RUN,
And learn from the listings,
But above all, have FUN!

Forget all the educational and technical stuff you've been reading in the rest of this magazine. Sure, it's been interesting and informative . . . but you need some fun too! Right? Relax then. You're in *my* territory now: Kelley's Korner—*the* place for great graphic games and sensational simulations.

## MAZE RACE

BY REGENA

Maze Race is a game written in TI BASIC for two players; one controls the red soldier, and one controls the blue soldier. The game starts out with the opposing soldiers lost at the ends of a forest maze. The object is to reach the safe zone across the field without meeting the enemy. The first soldier to cross his boundary into safety (through the entrance) wins the round, and the game continues until one soldier scores ten times. If the soldiers collide, neither one scores.

The maze is drawn randomly by the computer, so if an impossible maze is drawn (an entrance blocked or a soldier surrounded), it may be redrawn by answering the "Change Maze?" option with "Y" for yes.

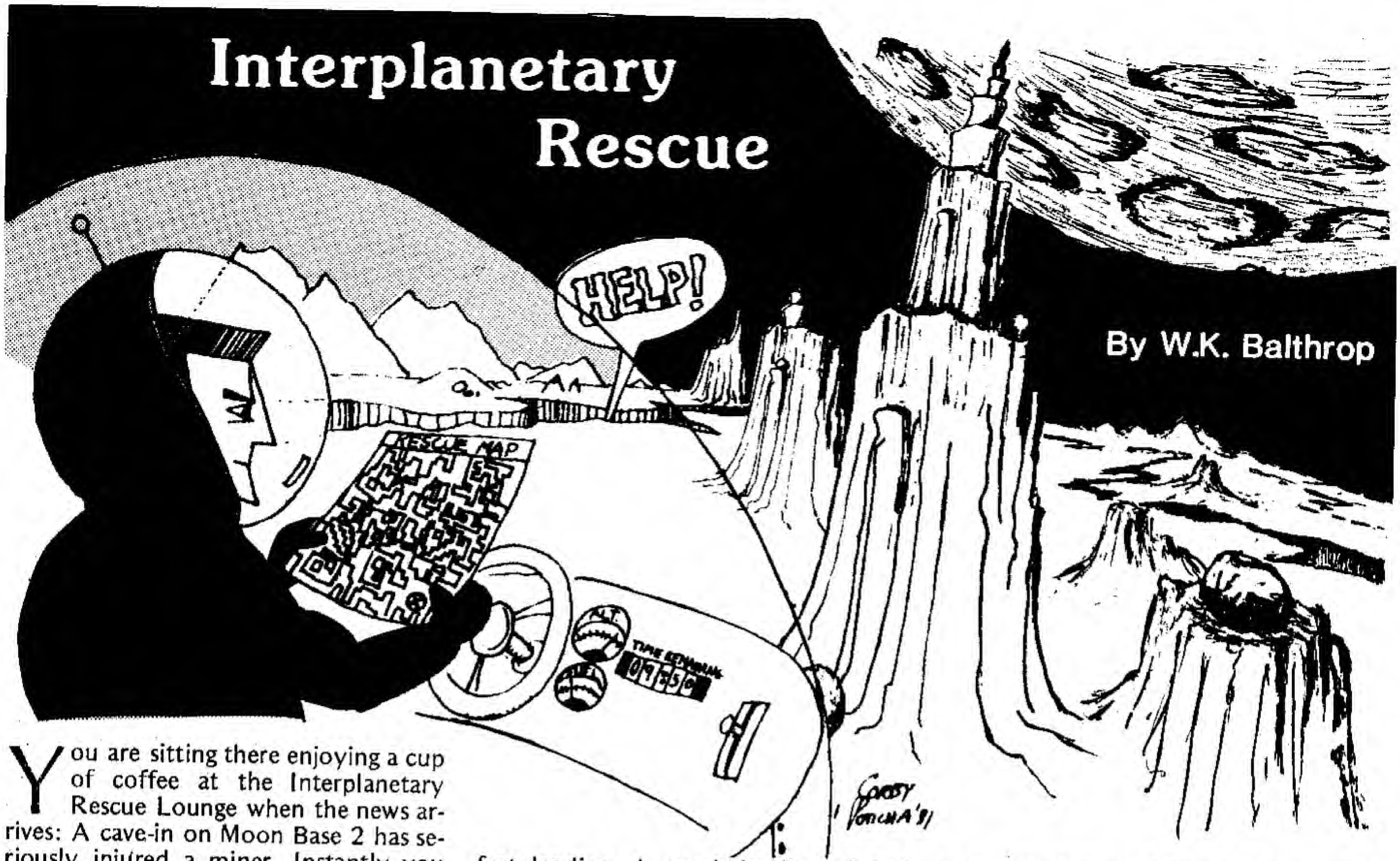
The red soldier is moved by pressing the arrow keys on the left keyboard. The blue soldier is moved by pressing I for up, J for left, K for right, and M for down. You may wish to use the *Video Games 1* Command Module overlay. No diagonal moves are allowed, and a soldier cannot go through a barrier. Once a key is pressed, the soldier moves in that direction until another key is pressed.

The difficulty of the maze may be altered by adjusting the PRINT statements 220-560. The & is a blank space on the maze, and # is a barrier.

Continued on p. 37

# Interplanetary Rescue

By W.K. Balthrop



You are sitting there enjoying a cup of coffee at the Interplanetary Rescue Lounge when the news arrives: A cave-in on Moon Base 2 has seriously injured a miner. Instantly you race for the shuttle, knowing you must reach the moon base, pick up the injured miner, and return to the base medical facilities. There's no time to waste!

Your ship is fueled and ready at the docking pad. In your ship you sit in front of your TI-99/4A controller panel and view the radar and instrumentation screen. You are now ready to take off. Press "T" on the control panel and the shuttle begins to lift. Using your six thrust control buttons you adjust your climb to the desired level. The terrain between you and Moon Base 2 is treacherous, and you must quickly ascertain the best route. Using your horizontal thrusters (arrow keys), you start your trip across the lunar landscape.

Accidents may happen anywhere, and right now your Interplanetary Rescue Team is in charge of the moon, Mars, and Venus. Use the arrow keys (E, S, D, X) to control horizontal movement. Horizontal velocity is listed on your control screen.

The elevations of the terrain show up as different colors on the map. At the right-hand side of the screen is a visual representation of your altitude in relation to the elevation colors. Your ship must be above the color on the right of the screen to safely pass through that color on the radar screen. Be careful not to overshoot the highest elevation color you plan to cross because it will waste valuable time getting back down and will burn precious fuel needed for the return voyage.

When you land on Moon Base 2, you must be traveling at a vertical speed of less than 6 meters/second to make a perfect landing. A rough landing of 6-10 meters/second will cause a leak in your main fuel valve, causing a loss of half your fuel. Any faster than 10 meters/second and you'll crash—never returning home. Once safely on the ground, the injured miner is put on board, and you're ready for the return trip. You won't have as much fuel holding you down, so it won't take as much thrust to accelerate vertically. Good luck on your rescue mission . . . you may need it!

### Instrument Displays:

ALT = altitude in meters. Each succeeding color on the radar represents 2000 meters.

HVEL = horizontal velocity across the radar screen (dependent upon arrow keys).

VVEL = vertical velocity; + is climbing, − is falling.

TIME = number of seconds into the mission.

FUEL = weight of fuel remaining in kilograms.

PWR = amount of thrust being generated. Each unit of thrust equals 1000 newtons; each newton equals approximately 2.05 kilograms of thrust.

## Calculations and variables

The gravity formula in Line 950 is the formula for speed of a falling object. V2 is the change in velocity in m/sec, F is the thrust in newtons, S is the weight of the ship in kg., E is the weight of remaining fuel in kg., and G is the gravity in m/sec$^2$. The time is one second in this calculation. All variables starting with D pertain to distance, and H is the altitude.

## Graphics

Characters accessible on the keyboard but not used in printing messages have been redefined (Lines 190-290). Then by using DISPLAY AT and a string, you can display colorful graphics very quickly without calling each square one-by-one. This method was used to display the radar map much more quickly than by using HCHAR and VCHAR. The strings are read in as 21 DATA statements. By changing the DATA statements in this program or adding some of your own, you may easily change the maps.

Sprites were used to depict the two crafts on the screen in order to be able to move with high resolution. Instead of using "CALL MOTION" which makes control of position difficult, I used "CALL LOCATE" which provides absolute control of the sprite's position.

| Option chosen | G | E | Take-off thrust |
|---|---|---|---|
| Moon | 2 | 20000 | 65000 |
| Mars | 4 | 45000 | 230000 |
| Venus | 6 | 80000 | 540000 |

**Thrust keys** T = initial thrust (displayed as 65, 230, or 540)

| | | | |
|---|---|---|---|
| units added | I +1 | O +5 | P +10 |
| units decreased | J −1 | K −5 | L −10 |

## GAMES & SIMULATIONS by FUTURA

| | |
|---|---|
| **ALL*STAR BASEBALL*** Pitching, fielding and base-running all under player control | $19.95 |
| **ALL*STAR BOWLING**** Tournament bowling at its best for up to eight contestants | $19.95 |
| **CHALLENGE I** Leaping frogs in two mind-challenging games | $16.95 |
| **CHALLENGE II** Nim and Tic-Tac-Toe with a new twist | $16.95 |
| **'CHUTES & SHARKS**** Rescue parachuters as they abandon ship over shark-infested ocean | $19.95 |
| **GALACTIC WAR**** Fast action space battle | $19.95 |
| **SAM DEFENSE** Command on authentic surface-to-air missile site | $19.95 |
| **WALL STREET*** Play the stock market and become a millionaire or a pauper | $19.95 |

*Also in Extended Basic
**Extended Basic only

**HOUSEHOLD INVENTORY SYSTEM**
**$34.95**

1st place

With this award-winning program you can accurately inventory personal property, calculate actual replacement value, and maintain records on a current basis

**ORDERING INFORMATION:** All orders shipped within 48 hours. Please enclose payment with order. Add $1.00 for postage. (Add $2.00 for postage on orders outside U.S.)

Ask for FUTURA Software at your local software dealer. Write for detailed description of these and other programs from FUTURA.

**Dealer Inquiries Invited**

**BUSINESS SOFTWARE**

| | |
|---|---|
| General Ledger | $99.95 |
| Accounts Payable | $99.95 |
| Accounts Receivable | $99.95 |
| Payroll | $99.95 |
| Inventory Management | $99.95 |
| Word Processing | $99.95 |
| Mailing List | $49.95 |

Fully interactive programs offer a totally integrated accounting system.
Loaded with innovative features:
— Completely menu-driven
— Random access of data
— Choice of retrieval of account records by name or number
— Fully prompted input
— Guaranteed to be bug-free and operator-proof!
— Hot-line and enhancement update service available.
Purchase units individually or the entire package for $500.00.
**Write for full details.**

**ANNOUNCING A SERIES OF EDUCATIONAL PROGRAMS**
**Use the power your computer has to focus**
***SIGHT *SOUND *ATTENTION *MOVEMENT**
**To help your child learn to read.**
**Write for full details and availability.**

FUTURA
S O F T W A R E

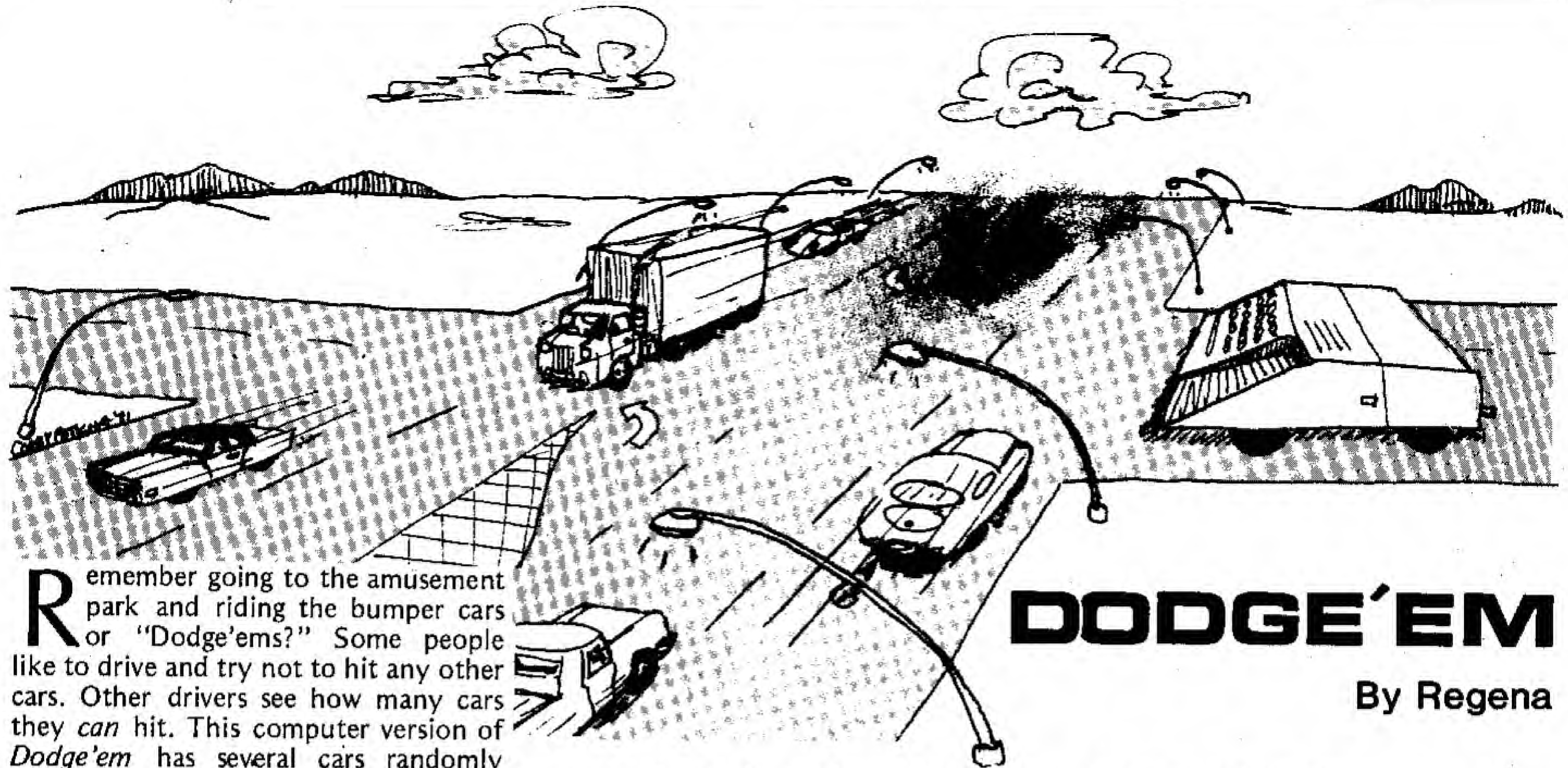
Ehninger Associates, Inc.
P.O. Box 5551
Fort Worth, Texas 76108
817/246-6536

## EXPLANATION OF THE PROGRAM
*Interplanetary Rescue*

| Line Nos. | |
|---|---|
| 170-310 | Clears screen; defines special characters and colors. |
| 320-350 | Initializes variables; branches to title screen and options. |
| 360-460 | Main control loop; GOSUB 790 receives the player's key presses. The VOL in the CALL SOUND statement depends upon the power. H is the altitude, and Line 400 tests for crashes. If the rescue craft has landed at either base, the program branches. |
| 470-670 | DATA statements to draw the "Novice" map. |
| 690-780 | Subroutine to label the parameters and draw the altimeter. |
| 790-1020 | Receives player's key responses and calculates parameters. |
| 1030-1090 | Prints updated altitude, time, velocities, fuel, and power. |
| 1100-1150 | Message and procedure for crashing into the hill. |
| 1160-1180 | Subroutine for procedure for any crash. |
| 1190-1260 | Prints score and option to play again; branches appropriately. |
| 1270-1400 | Messages and procedure for crashing at high velocity. |
| 1410-1460 | Procedure if craft lands safely; starts return trip. |
| 1470-1560 | Procedure for craft landing on return trip. |
| 1570-1620 | Prints and receives options of planet and level of difficulty. |
| 1630-1720 | Depending on the options chosen, sets gravity, fuel, and initial thrust, then prints appropriate map. |
| 1730-2350 | DATA statements for three maps. |
| 2360-2430 | Prints title screen. |

```
100 REM ***************************
110 REM * INTERPLANETARY RESCUE *
120 REM ***************************
130 REM 99'ER VERSION 1.4.1.XB
140 REM BY W.K. BALTHROP
150 REM
160 REM
170 CALL CLEAR
180 GOSUB 190 :: GOTO 320
190 CALL CHAR(96,"81423C3C3C3C4281")
200 CALL CHAR(62,"FF818199998181FF")
210 CALL CHAR(99,"026C9E1C24424201")
220 CALL CHAR(100,"00CC86C300107C20")
230 CALL CHAR(33,"FFFFFFFFFFFFFFFF")
240 CALL CHAR(94,"00")
250 CALL CHAR(95,"FFFFFFFFFFFFFFFF")
260 CALL CHAR(42,"FFFFFFFFFFFFFFFF")
270 CALL CHAR(63,"FFFFFFFFFFFFFFFF")
280 CALL CHAR(98,"183C3C7E7E7E66C3")
290 CALL CHAR(104,"5A5A5A185A")
300 CALL COLOR(1,4,1):: CALL COLOR(2,5,1)
310 RETURN
320 CALL SCREEN(16)
330 V=0 :: V1=0 :: V2=0 :: S=5000 :: F=0 :: H=0 :
    : D=0 :: T=0
340 TRIP=0 :: TIME=0 :: D1=0 :: D2=0 :: D3=25 :
    : D4=41 :: FF=0
350 GOSUB 1570
360 REM MAIN CONTROL LOOP
370 GOSUB 790 :: VOL=ABS((600000-F)/20000):: IF
    VOL>30 THEN VOL=30 ELSE IF VOL<0 THEN VOL=0
380 IF F>1 THEN CALL SOUND(-3000,110,VOL,220,
    VOL,110,VOL,-5,VOL)
390 CALL POSITION(#1,XC,YC):: CALL
    GCHAR(ABS((XC+4)/8+.5),ABS((YC+4)/8+.5),CC)
400 IF CC=94 AND H<2000 OR CC=95 AND H<4000 OR
    CC=42 AND H<6000 OR CC=63 AND H<8000 THEN 1100
410 IF TRIP=0 AND H>0 THEN TRIP=1
420 IF H=0 AND TRIP=1 THEN 1290
430 IF TRIP=2 AND H>0 THEN TRIP=3
440 IF TRIP=3 AND H=0 THEN 1470
450 TIME=TIME+1 :: IF H<=0 THEN V=0 :: V1=0
460 GOTO 370
```

Continued on p. 61



# DODGE'EM

By Regena

Remember going to the amusement park and riding the bumper cars or "Dodge'ems?" Some people like to drive and try not to hit any other cars. Other drivers see how many cars they *can* hit. This computer version of *Dodge'em* has several cars randomly moving up and down on the screen. The object of the game is to drive as quickly as you can from the right side of the screen to the left side of the screen. See what your mimimum time is for crossing. A short victory melody will be played if you cross successfully (no crashing). Of course, some of you players may tire of that and try to see how many crashes you can have in each crossing or within a certain time limit.

## Programming

My goal for this game was to make a game in Extended BASIC with as *short* a listing as possible so even the non-typing readers of *99'er Magazine* would not take too long to key-in a program to run. This program is a total of 73 statements yet contains 27 moving sprites. The actual game logic is contained in 21 lines (Lines 160 to 360). You could really have fewer lines by stacking statements if you don't mind long lines.

```
100 REM ************
110 REM * DODGE'EM *
120 REM ************
130 REM BY REGENA
140 REM 99'ER VERSION 1.4.1XB
150 GOTO 470
160 CALL CLEAR
170 CALL VCHAR(1,2,112,24):: CALL VCHAR(1,31,112,24)
180 DISPLAY AT(1,1):"TIME:"
190 RANDOMIZE
200 FOR I=2 TO 27
210 CALL SPRITE(#I,96,5,9,I*8+1,
    ((-1)^I)*INT(RND*12+1),0)
220 NEXT I
230 RV=0 :: CV=0 :: T=0 :: CR=0
240 CALL SPRITE(#1,104,7,INT(RND*180+1),
    233):: CALL SOUND(150,1397,0)
250 CALL KEY(0,K,S)
260 IF K=69 THEN RV=-5 :: CV=0
270 IF K=88 THEN RV=5 :: CV=0
280 IF K=83 THEN CV=-6 :: RV=0
290 IF K=68 THEN CV=0 :: RV=0
300 CALL MOTION(#1,RV,CV)
310 T=T+1 :: DISPLAY AT(1,7):T
320 CALL COINC(ALL,C):: IF C=0 THEN 340
330 CALL SOUND(150,-6,0):: CR=CR+1
340 CALL POSITION(#1,RO,CO):: IF CO>16 THEN 250
350 CALL MOTION(#1,0,0)
360 DISPLAY AT(3,1):"CRASHES: ";CR
370 IF CR>0 THEN 420
380 RESTORE 400
390 FOR I=1 TO 19 :: READ N :: CALL
    SOUND(100,N,1):: NEXT I
400 DATA 262,330,392,523,392,523,330,392,523,659
410 DATA 523,659,392,523,659,784,659,784,784
420 DISPLAY AT(24,1):"TRY AGAIN? (Y/N)"
430 CALL KEY(0,KE,S)
440 IF KE=89 THEN DISPLAY AT(3,1):"" :: GOTO 230
450 IF KE=78 THEN CALL CLEAR ELSE 430
460 STOP
470 CALL CLEAR :: CALL SOUND(500,-6,1,440,5)
480 DISPLAY AT(11,7):"D O D G E ' E M"
490 CALL SOUND(500,-7,1)
500 CALL CHAR(96,"387C7C38387C7C38")
510 CALL SOUND(500,-5,1)
520 CALL CHAR(104,"0066FFFFFF66")
530 CALL COLOR(11,11,11)
540 CALL CHAR(74,"10385492101010")
550 CALL CHAR(81,"08080808492A1C08")
560 CALL CHAR(88,"102040FF40201")
570 CALL CHAR(90,"080402FF020408")
580 CALL CLEAR
590 DISPLAY AT(2,7):"D O D G E ' E M"
600 DISPLAY AT(5,2):"DRIVE THE RED CAR FROM THE"
610 DISPLAY AT(6,2):"RIGHT SIDE OF THE SCREEN"
620 DISPLAY AT(7,2):"TO THE LEFT SIDE BY USING"
630 DISPLAY AT(8,2):"THE ARROW KEYS J Q X Z."
640 DISPLAY AT(10,2)BEEP:"TRY NOT TO CRASH INTO"
650 DISPLAY AT(11,2):"OTHER CARS."
660 DISPLAY AT(13,2):"YOU MAY SLOW DOWN OR STOP"
670 DISPLAY AT(14,2):"BY PRESSING Z BUT YOU MAY"
680 DISPLAY AT(15,2):"NOT BACK UP."
690 DISPLAY AT(24,2):"PRESS ANY KEY TO START."
700 CALL KEY(0,KEY,S):: IF S=0 THEN 700 ELSE 160
710 IF S=0 THEN 700 ELSE 160
720 END
```

### EXPLANATION OF THE PROGRAM *DODGE'EM*

| Line Nos. | |
|---|---|
| 100-150 | Introductory REMarks; branches to title screen. |
| 160 | Clears screen. |
| 170-180 | Draws left and right borders; prints "TIME:" |
| 190-220 | Places 26 cars moving vertically at random speeds. |
| 230-240 | Initializes variables; randomly places red car at right side of screen; beeps. |
| 250-300 | Depending on key pressed, sets row velocity and column velocity and moves red car. |
| 310 | Increments and prints time counter. |
| 320 | Checks coincidence for a crash. |
| 330 | If there is a crash, sounds a crashing noise and increments number of crashes. |
| 340 | If the car is not at the left border, program branches to Line 240. |
| 350-360 | Stops the red car and prints the number of crashes. |
| 370-410 | If there were no crashes, plays victory melody. |
| 420-460 | Asks if player wants to try again and branches appropriately. |
| 470-570 | Prints title screen while sounding crashing noises and defining special characters. |
| 580-720 | Prints instructions. |

Maze Race . . . from p. 33

## EXPLANATION OF THE PROGRAM
*Maze Race*

| Line Nos. | |
|---|---|
| 170 | Branches to title screen and instructions. |
| 180-210 | Subroutine to print messages on screen below maze. |
| 220-570 | Subroutines to print maze a line at a time. |
| 580-700 | Clears screen and prints maze. Lines of maze are chosen randomly then printed. |
| 710-740 | Places soldiers at opposite ends of maze in random horizontal position. |
| 750-810 | Prints message, "CHANGE MAZE?", waits for response and branches accordingly. |
| 820-900 | Initializes variables. RX, RY, BX, and BY are directional increments. RXC, RYC, BXC, and BYC are coordinates for the red and blue soldiers. RED and BLUE = 1 for a win, 0 for a loss. Sounds a "beep" to start game. |
| 910-1070 | Reads red soldier's keyboard entry to move. |
| 1080-1230 | Checks where soldier will move and redraws soldier. Checks location for space, block, enemy entrance, or his goal. |
| 1240-1400 | Reads blue soldier's keyboard entry to move. |
| 1410-1580 | Checks blue soldier's move and location. |
| 1590-1690 | Routine if soldiers collide. |
| 1700-1760 | Prints message when one soldier wins. |
| 1770-1940 | Prints scores. |
| 1950-2000 | Asks "TRY AGAIN?" and branches accordingly. |
| 2010-2180 | Prints title screen and defines characters and colors; asks if instructions are needed and waits for response. |
| 2190-2270 | Prints instructions if desired. |

```
100 REM  *************
110 REM  * MAZE RACE *
120 REM  *************
130 REM
140 REM  BY REGENA
150 REM   99'ER VERSION 1.4.1
160 REM
170 GOTO 2010
180 FOR J=1 TO LEN(M$)
190 CALL HCHAR(X,J,ASC(SEG$(M$,J,1)))
200 NEXT J
210 RETURN
220 PRINT "&##&#&#&&&&&&#&#&&&&&###&#&"
230 RETURN
240 PRINT "&####&&#&&&&#&&#&&#&&#&&&&&"
250 RETURN
260 PRINT "&&&&&&#&#&#&&#&&&&#&&&&#&&#"
270 RETURN
280 PRINT "&#&&&#&&&&&&&&&#&&&&&#&&&&&#"
290 RETURN
300 PRINT "&&&#&&##&&&#&&#&&#&&&&##&&#&&"
310 RETURN
320 PRINT "&&##&##&##&&&&&&#&&&&&&&#&&&"
330 RETURN
340 PRINT "#&&#&&##&#&&&&&&&&##&&&####&&"
350 RETURN
360 PRINT "&#&&&#&&&#&&###&##&&&#&&&&#"
370 RETURN
380 PRINT "&&#&&&&&#&&&&&&&#&&&#&&&&&#&"
390 RETURN
400 PRINT "&&##&&&&##&&&&&&##&&&#&&&&&&"
410 RETURN
420 PRINT "#&#&&&##&&&&##&&&&&&#&&#&&&&#"
430 RETURN
440 PRINT "#&&&#&&&&#&&&&&#&&&&&&#&&&#&&"
450 RETURN
460 PRINT "&#&&&#&&&&&#&&&&&&&&#&&&&##"
470 RETURN
480 PRINT "&&#&&&&#&&&&&&#&&&#&&&&#&&&"
490 RETURN
500 PRINT "#&&&#&&&#&#&#&&&&&&&#&&&&##&"
510 RETURN
520 PRINT "&##&&&##&&&&&###&&#&&&#&&&&&"
530 RETURN
540 PRINT "&&&#&&&&&#&&&&&&#&&&&&&#&&&&"
550 RETURN
560 PRINT "&&&&##&&&&##&&&#&&&&&##&&&&&&"
570 RETURN
580 CALL CLEAR
590 RANDOMIZE
600 CALL HCHAR(23,3,35,28)
610 FOR I=2 TO 20
620 AA=INT(18*RND)+1
630 ON AA GOSUB 220,240,260,280,300,320,
    340,360,380,400,420,440,460,480,500,
    520,540,560
640 NEXT I
650 PRINT :::
660 CALL HCHAR(21,3,35,28)
670 CALL VCHAR(1,2,113,21)
680 CALL VCHAR(1,30,105,21)
690 CALL VCHAR(4,2,38,3)
700 CALL VCHAR(16,30,38,3)
710 RXC=INT(18*RND)+2
720 CALL HCHAR(RXC,3,104)
730 BXC=INT(18*RND)+2
740 CALL HCHAR(BXC,29,112)
750 M$="CHANGE MAZE? (Y/N)  "
760 X=24
770 GOSUB 180
780 CALL KEY(0,KEY.S)
790 IF KEY=89 THEN 580
800 IF KEY<>78 THEN 780
810 CALL HCHAR(24,1,32,19)
820 RX=0
830 RY=0
840 BX=0
850 BY=0
860 RYC=3
870 BYC=29
880 RED=0
890 BLUE=0
900 CALL SOUND(150,1397,2)
910 CALL KEY(1,K1,S1)
920 IF S1=0 THEN 1080
930 IF K1<>5 THEN 970
940 RX=-1
950 RY=0
960 GOTO 1080
970 IF K1<>3 THEN 1010
980 RY=1
990 RX=0
1000 GOTO 1080
1010 IF K1+1<>1 THEN 1050
1020 RX=1
1030 RY=0
1040 GOTO 1080
1050 IF K1<>2 THEN 1080
1060 RY=-1
1070 RX=0
1080 IF RYC+RY>1 THEN 1120
1090 M$="RED WENT WRONG
     WAY-BLUE WON."
1100 BLUE=1
1110 GOTO 1440
1120 CALL GCHAR(RXC+RX,
     RYC+RY,CC)
1130 IF CC=38 THEN 1170
1140 IF CC=112 THEN 1590
1150 CALL SOUND(150,-5,0)
1160 GOTO 1240
1170 CALL HCHAR(RXC,RYC,38)
1180 RXC=RXC+RX
1190 RYC=RYC+RY
1200 CALL HCHAR(RXC,RYC,104)
1210 IF RYC<>30 THEN 1240
1220 RED=1
1230 GOTO 1700
1240 CALL KEY(2,K2,S2)
1250 IF S2=0 THEN 1410
1260 IF K2<>5 THEN 1300
1270 BX=-1
1280 BY=0
1290 GOTO 1410
1300 IF K2<>3 THEN 1340
1310 BY=1
1320 BX=0
1330 GOTO 1410
1340 IF K2+1<>1 THEN 1380
1350 BX=1
1360 BY=0
1370 GOTO 1410
1380 IF K2<>2 THEN 1410
1390 BY=-1
1400 BX=0
1410 IF BYC+BY<31 THEN 1470
1420 M$="BLUE WENT WRONG
     WAY-RED WON."
1430 RED=1
1440 X=22
1450 GOSUB 180
1460 GOTO 1770
1470 CALL GCHAR(BXC+BX,BYC+BY,DD)
1480 IF DD=38 THEN 1520
1490 IF DD=104 THEN 1590
1500 CALL SOUND(150,-7,0)
1510 GOTO 910
1520 CALL HCHAR(BXC,BYC,38)
1530 BXC=BXC+BX
1540 BYC=BYC+BY
1550 CALL HCHAR(BXC,BYC,112)
1560 IF BYC<>2 THEN 910
1570 BLUE=1
1580 GOTO 1700
1590 CALL HCHAR(BXC,BYC,38)
1600 CALL HCHAR(RXC,RYC,120,2)
1610 CALL HCHAR(RXC+RX,RYC+RY,
     120,2)
1620 FOR 1=1 TO 15
1630 CALL COLOR(12,7,6)
1640 CALL COLOR(12,6,7)
1650 NEXT I
1660 M$="DUEL'' BOTH SOLDIERS
     DIE."
1670 X=23
1680 GOSUB 180
1690 GOTO 1930
1700 CALL HCHAR(23,1,32,64)
1710 IF BLUE=1 THEN 1740
1720 M$="RED WINS THIS TIME."
1730 GOTO 1750
1740 M$="BLUE WINS THIS TIME."
1750 X=22
1760 GOSUB 180
1770 M$="TOTAL
     SCORE:  RED      BLUE "
1780 X=23
1790 GOSUB 180
1800 REDS=REDS+RED
1810 BLUES=BLUES+BLUE
1820 IF REDS<10 THEN 1860
1830 CALL HCHAR(23,19,49)
1840 CALL HCHAR(23,20,48)
1850 GOTO 1910
1860 CALL HCHAR(23,19,REDS+48)
1870 IF BLUES<10 THEN 1910
1880 CALL HCHAR(23,29,49)
1890 CALL HCHAR(23,30,48)
1900 GOTO 2000
1910 CALL HCHAR(23,29,BLUES+48)
1920 IF REDS=10 THEN 2000
1930 FOR DELAY=1 TO 1000
1940 NEXT DELAY
1750 PRINT "TRY AGAIN? (Y/N)"
1960 CALL KEY(0,KEY,S)
1970 IF S=0 THEN 1960
1980 IF KEY=89 THEN 580
1990 IF KEY<>78 THEN 1960
2000 STOP
2010 CALL CLEAR
2020 PRINT TAB(7);"*************":TAB(7);"*           *"
2030 PRINT TAB(7);"* MAZE RACE *":TAB(7);"*           *"
2040 PRINT TAB(7);"*************"::::::::::
2050 CALL CHAR(38,"0")
2060 CALL CHAR(35,"FFFFFFFFFFFFFFFF")
2070 CALL COLOR(1,13,1)
2080 CALL CHAR(104,"18187E1818242424")
2090 CALL CHAR(105,"FFFFFFFFFFFFFFFF")
2100 CALL COLOR(10,7,1)
2110 CALL CHAR(120,"49221449142249")
2120 CALL CHAR(112,"18187E1818242424")
2130 CALL CHAR(113,"FFFFFFFFFFFFFFFF")
2140 CALL COLOR(11,6,1)
2150 PRINT "WANT INSTRUCTIONS? (Y/N)"
2160 CALL KEY(0,KEY,S)
2170 IF KEY=78 THEN 580
2180 IF KEY<>89 THEN 2160
2190 CALL CLEAR
2200 PRINT "TWO OPPOSING SOLDIERS ARE":"LOST IN A JUNGLE MAZE."
2210 PRINT :"EACH MAN MUST FIND HIS WAY":"TO THE OPPOSITE BORDER"
2220 PRINT "TO BE SAFE."::"DO NOT COLLIDE OR ELSE"
2230 PRINT "BOTH SOLDIERS WILL DIE."
2240 PRINT :::"PRESS ENTER TO START."
2250 CALL KEY(0,KEY,S)
2260 IF KEY=13 THEN 580 ELSE 2250
2270 END
```

**SUPPORT OUR ADVERTISERS. THEY TOO MAKE THIS MAGAZINE POSSIBLE . . .**

**MOVING?**

**Don't Miss Out On Any Issues Of 99'er Magazine**

Send us a Change-of-Address Card (available at any Post Office) 6–8 weeks prior to the move.

Be sure to include both the old & new address, plus the alphanumeric code above your name on the mailing label.

**ATTENTION**

**BOOKSTORES, DEALERS, & DISTRIBUTORS**

**CALL OR WRITE US ABOUT SELLING SINGLE COPIES OF**

99'er magazine™

# HUGE ELEK-TEK DISCOUNTS ON TI-99/4A Home Computer System

TEXAS INSTRUMENTS INCORPORATED

| Model | Name | Mfr. Sugg. Ret. | Elek-Tek Price |
|---|---|---|---|
| **CONSOLE** | | | |
| PHC 004A | TI-99/4A Home Computer | 525.00 | 360.00 |
| **PERIPHERALS** | | | |
| PHP 1500 | Solid State Speech™ Synthesizer | 149.95 | 125.00 |
| PHP 1600 | Telephone Coupler (Modem) | 224.95 | 185.00 |
| PHP 1700 | RS-232 Accessories Interface | 224.95 | 185.00 |
| PHP 1800 | Disk Drive Controller (One Disk Manager module packed with each Disk Controller) | 299.95 | 238.00 |
| PHP 1850 | Disk Memory Drive | 499.95 | 395.00 |
| PHP 1900 | Solid State Printer | 399.95 | 300.00 |
| PHP 2200 | Memory Expansion (32K RAM) | 399.95 | 300.00 |
| PHP 2400 | P-Code Peripheral | 399.95 | 300.00 |
| PHA 2100 | R.F. Modulator (TV Adapter) | 49.95 | 40.00 |
| PHA 4100 | 10" Color Monitor | 399.95 | 330.00 |
| **OPTIONAL ACCESSORIES** | | | |
| PHP 1100 | Wired Remote Controllers (Pair) | 34.95 | 28.00 |
| PHA 1950 | Thermal Paper (2 Pack) | 9.95 | 8.50 |
| PHA 2000 | Dual Cassette Cable | 14.95 | 11.75 |
| PHA 2010 | Monitor Cable | 19.95 | 16.00 |
| PHA 2020 | Audio Adapter (Headphone Jack) | 19.95 | 16.00 |
| PHA 2605 | Blank Overlays (4 Pack) | 7.95 | 6.50 |
| **APPLICATION PROGRAMS** | | | |
| **Home Management/Personal Finance** | | | |
| | **Command Modules** | | |
| PHM 3006 | Home Financial Decisions | 29.95 | 25.45 |
| PHM 3007 | Household Budget Management (Data storage system is required) | 39.95 | 33.95 |
| PHM 3012 | Securities Analysis | 54.95 | 46.70 |
| PHM 3013 | Personal Record Keeping (Data storage system is recommended) | 49.95 | 42.45 |
| PHM 3016 | Tax/Investment Record Keeping (Disk system is required) | 69.95 | 59.45 |
| PHM 3022 | Personal Real Estate (Data storage system is recommended) | 69.95 | 59.45 |
| PHM 3044 | Personal Report Generator (Data storage system is recommended) | 49.95 | 42.45 |
| | **Diskette** | | |
| PHD 5001 | Mailing List | 69.95 | 59.45 |
| PHD 5003 | Personal Financial Aids | 19.95 | 16.95 |
| PHD 5021 | Checkbook Manager | 19.95 | 16.95 |
| PHD 5022 | Business Aids Library — Finance Management (Extended BASIC Command Module is required) | 39.95 | 33.95 |
| PHD 5024 | Business Aids Library — Inventory Management (Personal Record Keeping or Statistics Command Module is required) | 69.95 | 59.45 |
| PHD 5027 | Business Aids Library — Invoice Management (Personal Record Keeping or Statistics Command Module is required) | 69.95 | 59.45 |
| PHD 5029 | Business Aids Library — Cash Management (Extended BASIC Command Module is required) | 39.95 | 33.95 |
| PHD 5038 | Business Aids Library — Lease/Purchase Decisions | 39.95 | 33.95 |
| | **Cassette** | | |
| PHT 6003 | Personal Financial Aids | 14.95 | 12.70 |
| PHT 6038 | Business Aids Library — Lease/Purchase Decisions | 34.95 | 28.00 |
| **Education/Personal Enrichment** | | | |
| | **Command Modules** | | |
| PHM 3002 | Early Learning Fun | 29.95 | 25.45 |
| PHM 3003 | Beginning Grammar | 29.95 | 25.45 |
| PHM 3004 | Number Magic | 19.95 | 16.95 |
| PHM 3005 | Video Graphs | 19.95 | 16.95 |
| PHM 3008 | Video Chess | 69.95 | 59.45 |
| PHM 3010 | Physical Fitness | 29.95 | 25.45 |
| PHM 3015 | Early Reading† (Solid State Speech™ Synthesizer is required) | 54.95 | 46.70 |
| PHM 3020 | Music Maker (Data storage system is recommended) | 39.95 | 33.95 |
| PHM 3021 | Weight Control and Nutrition (Data storage system is recommended) | 59.95 | 50.95 |
| PHM 3027 | Addition and Subtraction I† (Solid State Speech™ Synthesizer is recommended) | 39.95 | 33.95 |
| PHM 3028 | Addition and Subtraction II† (Solid State Speech™ Synthesizer is recommended) | 39.95 | 33.95 |
| PHM 3029 | Multiplication I† (Solid State Speech™ Synthesizer is recommended) | 39.95 | 33.95 |
| PHM 3040 | TI LOGO (Memory Expansion is required) | 299.95 | 200.00 |
| PHM 3043 | Reading Fun† | 54.95 | 46.70 |
| | **Diskette** | | |
| PHD 5007 | Teach Yourself BASIC | 34.95 | 29.70 |
| PHD 5009 | Music Skills Trainer | 29.95 | 25.45 |
| PHD 5011 | Computer Music Box | 19.95 | 16.95 |
| PHD 5018 | Market Simulation | 19.95 | 16.95 |
| PHD 5019 | Teach Yourself Extended BASIC (Extended BASIC Command Module is required) | 24.95 | 21.20 |
| PHD 5020 | Music Maker Demonstration (Music Maker Command Module is required) | 14.95 | 12.70 |
| PHD 5023 | Basketball Statistics (Extended BASIC Command Module is required) | 24.95 | 21.20 |
| PHD 5026 | Bridge Bidding I | 29.95 | 25.45 |
| PHD 5030 | Speak & Spell™ Program (Solid State Speech™ Synthesizer is required) | 29.95 | 25.45 |
| PHD 5031 | Speak & Math™ Program (Solid State Speech™ Synthesizer and Terminal Emulator II are required) | 29.95 | 25.45 |
| PHD 5039 | Bridge Bidding II | 29.95 | 25.45 |
| PHD 5041 | Bridge Bidding III | 29.95 | 25.45 |
| PHD 5042 | Spell Writer (Terminal Emulator II Command Module and Solid State Speech™ Synthesizer are required) | 29.95 | 25.45 |
| | **Cassette** | | |
| PHT 6007 | Teach Yourself BASIC | 29.95 | 25.45 |
| PHT 6009 | Music Skills Trainer | 24.95 | 21.20 |
| PHT 6011 | Computer Music Box | 14.95 | 12.70 |
| PHT 6018 | Market Simulation | 14.95 | 12.70 |
| PHT 6019 | Teach Yourself Extended BASIC (Extended BASIC Command Module is required) | 19.95 | 16.95 |
| PHT 6026 | Bridge Bidding I | 24.95 | 21.20 |
| PHT 6031 | Speak & Math™ Program (Solid State Speech™ Synthesizer and Terminal Emulator II are required) | 24.95 | 21.20 |
| **APPLICATION PROGRAMS CONTINUED** | | | |
| **Education/Personal Enrichment Continued** | | | |
| PHT 6039 | Bridge Bidding II | 24.95 | 21.20 |
| PHT 6041 | Bridge Bidding III | 24.95 | 21.20 |
| PHT 6042 | Spell Writer (Terminal Emulator II Command Module and Solid State Speech™ Synthesizer are required) | 24.95 | 21.20 |
| **Entertainment** | | | |
| | **Command Modules** | | |
| PHM 3009 | Football | 29.95 | 25.45 |
| PHM 3018 | Video Games I* | 29.95 | 25.45 |
| PHM 3023 | Hunt the Wumpus | 24.95 | 21.20 |
| PHM 3024 | Indoor Soccer | 29.95 | 25.45 |
| PHM 3025 | Mind Challengers | 24.95 | 21.20 |
| PHM 3030 | A-Maze-Ing | 24.95 | 21.20 |
| PHM 3031 | The Attack†† | 39.95 | 33.95 |
| PHM 3032 | Blasto†† | 24.95 | 21.20 |
| PHM 3033 | Blackjack and Poker†† | 24.95 | 21.20 |
| PHM 3034 | Hustle†† | 24.95 | 21.20 |
| PHM 3036 | ZeroZap†† | 19.95 | 16.95 |
| PHM 3037 | Hangman†† | 19.95 | 16.95 |
| PHM 3038 | Connect Four†† | 19.95 | 16.95 |
| PHM 3039 | Yahtzee†† | 24.95 | 21.20 |
| PHM 3041D | Adventure (Pirate Adventure Diskette Game Included) | 49.95 | 42.45 |
| PHM 3041T | Adventure (Pirate Adventure Cassette Game Included) | 49.95 | 42.45 |
| PHM 3052 | Tombstone City: 21st Century | 39.95 | 33.95 |
| PHM 3053 | TI Invaders | 39.95 | 33.95 |
| PHM 3054 | Car Wars | 39.95 | 33.95 |
| PHM 3057 | Munch Man **(available 2nd quarter)** | 39.95 | 33.95 |
| | **Diskette** | | |
| PHD 5002 | TI-Trek (with speech) | 14.95 | 12.70 |
| PHD 5010 | Mystery Melody | 14.95 | 12.70 |
| PHD 5015 | Oldies But Goodies — Games I | 19.95 | 16.95 |
| PHD 5017 | Oldies But Goodies — Games II | 24.95 | 21.20 |
| PHD 5025 | Saturday Night Bingo (Solid State Speech™ Synthesizer is required) | 29.95 | 25.45 |
| PHD 5037 | Draw Poker | 24.95 | 21.20 |
| | Adventure Series | | |
| | **The following adventure diskettes require the PHM 3041 D Command Module** | | |
| PHD 5046 | Adventureland | 29.95 | 25.45 |
| PHD 5047 | Mission Impossible | 29.95 | 25.45 |
| PHD 5048 | Voodoo Castle | 29.95 | 25.45 |
| PHD 5049 | The Count | 29.95 | 25.45 |
| PHD 5050 | Strange Odyssey | 29.95 | 25.45 |
| PHD 5051 | Mystery Fun House | 29.95 | 25.45 |
| PHD 5052 | Pyramid of Doom | 29.95 | 25.45 |
| PHD 5053 | Ghost Town | 29.95 | 25.45 |
| PHD 5054 | Savage Island I & II | 39.95 | 33.95 |
| PHD 5056 | Golden Voyage | 29.95 | 25.45 |
| | **Cassette** | | |
| PHT 6010 | Mystery Melody | 9.95 | 8.50 |
| PHT 6015 | Oldies But Goodies — Games I | 14.95 | 12.70 |
| PHT 6017 | Oldies But Goodies — Games II | 19.95 | 16.95 |
| PHT 6025 | Saturday Night Bingo (Solid State Speech™ Synthesizer is required) | 24.95 | 21.20 |
| PHT 6037 | Draw Poker | 19.95 | 16.95 |
| | Adventure Series | | |
| | **The following adventure cassettes require the PHM 3041 T Command Module** | | |
| PHT 6046 | Adventureland | 29.95 | 25.45 |
| PHT 6047 | Mission Impossible | 29.95 | 25.45 |
| PHT 6048 | Voodoo Castle | 29.95 | 25.45 |
| PHT 6049 | The Count | 29.95 | 25.45 |
| PHT 6050 | Strange Odyssey | 29.95 | 25.45 |
| PHT 6051 | Mystery Fun House | 29.95 | 25.45 |
| PHT 6052 | Pyramid of Doom | 29.95 | 25.45 |
| PHT 6053 | Ghost Town | 29.95 | 25.45 |
| PHT 6054 | Savage Island I & II | 39.95 | 33.95 |
| PHT 6056 | Golden Voyage | 29.95 | 25.45 |
| **OTHER APPLICATION PROGRAMS** | | | |
| | **Command Modules** | | |
| PHM 3000 | Diagnostic | 29.95 | 25.45 |
| PHM 3001 | Demonstration | 69.95 | 59.45 |
| PHM 3011 | Speech Editor (Solid State Speech™ Synthesizer is required) | 44.95 | 38.20 |
| PHM 3014 | Statistics (Data storage system is recommended) | 44.95 | 38.20 |
| PHM 3026 | Extended BASIC | 99.95 | 75.00 |
| PHM 3035 | Terminal Emulator II | 49.95 | 42.45 |
| PHM 3055 | Editor/Assembler* | 99.95 | 84.95 |
| PHM 3058 | Mini-Memory (4K) | 99.95 | 84.95 |
| | **Diskette** | | |
| PHD 5004 | Programming Aids I | 14.95 | 12.70 |
| PHD 5005 | Programming Aids II | 24.95 | 21.20 |
| PHD 5006 | Math Routine Library | 29.95 | 25.45 |
| PHD 5008 | Electrical Engineering Library | 29.95 | 25.45 |
| PHD 5012 | Programming Aids III | 19.95 | 16.95 |
| PHD 5013 | Graphing Package | 19.95 | 16.95 |
| PHD 5016 | Structural Engineering Library | 29.95 | 25.45 |
| | **Cassette** | | |
| PHT 6004 | Programming Aids I | 9.95 | 8.50 |
| PHT 6006 | Math Routine Library | 24.95 | 21.20 |
| PHT 6008 | Electrical Engineering Library | 24.95 | 21.20 |
| PHT 6013 | Graphing Package | 14.95 | 12.70 |
| PHT 6016 | Structural Engineering Library | 24.95 | 21.20 |
| **SOFTWARE LIBRARIES** | | | |
| PHL 7001 | The Home Financial Manager | 139.95 | 118.95 |
| PHL 7002 | The Family Entertainer | 89.95 | 76.45 |
| PHL 7003 | The Elementary Educator | 99.95 | 84.95 |
| PHL 7004 | The Music Educator | 64.90 | 55.20 |
| PHL 7005 | The Super Programmer | 119.00 | 101.20 |
| PHL 7006 | The Speaking Math Teacher | 119.85 | 101.95 |
| PHL 7007 | The Speaking Reading Teacher | 109.90 | 93.45 |
| PHL 7009 | The TI Arcade Game Series | 119.85 | 101.95 |
| PHL 7010 | The Milton Bradley Game Series | 114.75 | 97.45 |
| PHL 7011 | The Computer Introductory Package | 119.85 | 101.95 |

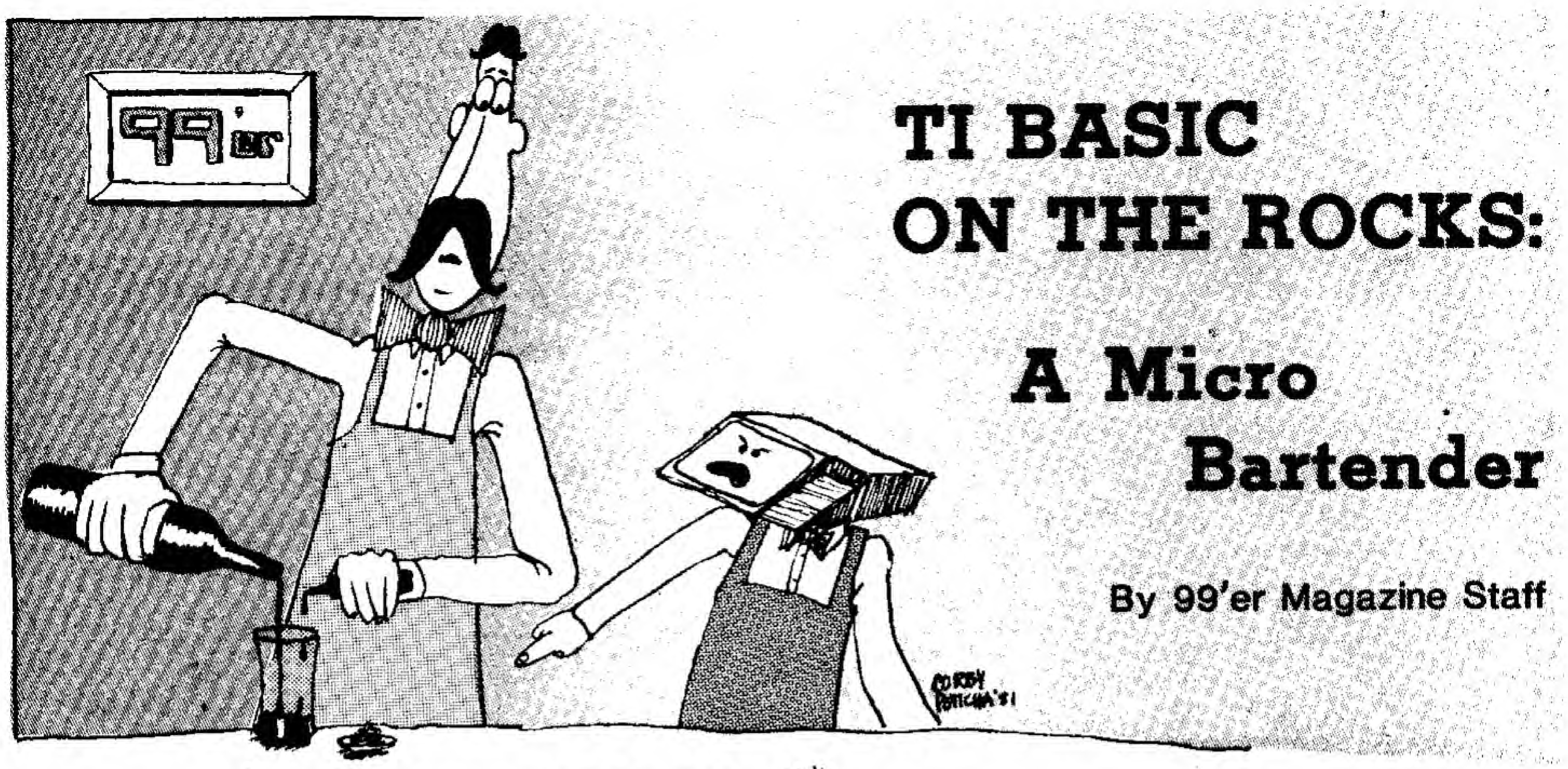
†Developed by Scott Foresman    ††Developed by Milton Bradley — The Attack, Blasto, Hustle, ZeroZap, Connect Four and Yahtzee are trademarks of Milton Bradley

**ELEK-TEK, inc.**

6557 North Lincoln Avenue, Chicago IL 60646
(800) 621-1269    (312) 677-7660

**CALL TOLL FREE 800-621-1269 (EXCEPT Illinois, Alaska, Hawaii)**

Accessories discounted too. **Corporate Accounts Invited.** Mastercharge or Visa by mail or phone. Mail Cash, Ck., Mon. Ord., Pers. Ck. [2 wks to clear]. Add $4.00 1st item; $1.00 ea. add'l shpg. & handl. Shipments to IL address add 6% tax. Prices subject to change. **WRITE for free catalog. ALL ELEK-TEK MERCHANDISE IS BRAND NEW, FIRST QUALITY AND COMPLETE.**



# TI BASIC ON THE ROCKS: A Micro Bartender

By 99'er Magazine Staff

"...AND DON'T FORGET THE ICE!!"

Entertaining guests can indeed be a chore—especially when you have to help them decide on their choice of drinks, remember how to correctly mix the selected drinks, and simultaneously explain to your curious visitors exactly how you use that exotic computer in your livingroom. Now, this three-part task can be handled much more enjoyably with *Bartender*—a TI BASIC program.

The next time guests arrive just sit them in front of your home computer and let them choose their own mixed drinks. The program will not only provide easy-to-follow recipes, but will also show your guests how the finished drinks should appear—in full color, with proper glass and garnish!

But what's the use of choosing drinks that are impossible to make because you're missing one or more ingredients? It's definitely slow and frustrating when the only way to find "possible" drinks is by scanning all the ingredients on page after page of recipes. But alas, this tedious process is now a thing of the past . . . With *Bartender's* built-in search routine, you can first tell the computer what ingredients are actually on hand, and it will tell you what drinks you can, in fact, make. Then, you can look up the details of each recipe and see a graphic representation of how the finished drink is supposed to look.

Cramming nearly a score of drink recipes (plus their associated graphics) into the TI-99/4's 16K of RAM memory was no easy feat . . . Observant programmers will notice our extensive use of data reconstruction techniques. And for those programmers who happen to be non-drinkers—indeed a rarity these days with program debugging often credited with "driving a man to drink . . . " —the program logic and control structure is suitable with many other types of reconstructed "recipes." [Only kidding, of course, about the "driving a man to drink . . . "—Ed.]

EXPLANATION OF THE PROGRAM
*Bartender*

| Line Nos. | |
|---|---|
| 200-240 | Prints title screen. |
| 250-290 | Subroutine to determine colors for graphics. |
| 300-1350 | Subroutine for graphics. |
| 1360-1650 | Defines special characters. |
| 1660-1750 | Reads data while title screen is displayed. |
| 1760-1860 | Prints screen of two major options. |
| 1870-2220 | First option. Prints two menu screens of the list of drinks, receives user's choice. |
| 2230-2250 | Clears screen, sets colors for graphics for drink chosen. |
| 2260-2540 | Prints name of drink and type of glass. |
| 2550-2580 | Prints amounts and ingredients in recipe. |
| 2590-2650 | Prints mixing instructions. |
| 2660-2710 | Prints cocktail or whiskey sour glass. |
| 2720-2810 | Prints garnish and sets colors for garnish. |
| 2820-2850 | Prints instructions for stir rod or straws. |
| 2860-3000 | Draws the drink. |
| 3010-3020 | User may press any key to continue program. |
| 3030-3090 | Second option. Prints instructions for ingredient inventory. |
| 3100-3200 | Receives user's input Y or N for each ingredient in INV$ array. |
| 3220-3260 | Prints message for no drinks possible. |
| 3270-3370 | Compares each drink's ingredients with inventory list and prints possible drinks to make. |
| 3380-3400 | User presses any key to go back to option screen. |
| 3410-3780 | Data for DRINK$ array of attributes for each drink. |
| 3790-3810 | Names of ingredients for inventory list. |

```
100 REM  ** BARTENDER **
110 REM
120 REM  DESIGN BY G.M. KAPLAN
130 REM  CODING BY MARK MOSELEY
140 REM  EDITING BY W.K. BALTHROP
150 REM  REVISION 2.0 BY CHERYL WHITELAW
160 REM  99'ER VERSION 1.4.2
170 CALL SCREEN(8)
180 CALL CLEAR
190 DIM DRINK$(19,23),INV$(15,1)
200 PRINT "** BARTENDER **"::::::::::
210 CALL COLOR(2,7,16)
220 C=7
230 GOSUB 250
240 GOTO 1360
250 CALL COLOR(14,13,C)
260 CALL COLOR(11,2,C)
270 CALL COLOR(16,11,C)
```

Continued on p. 84

# Notes on a Computer Score:

## Part 1 - The TI-99/4 Conducts Music Theory Drill in a Traditional Classroom Setting.

By Norma and John Clulow

Recently I returned to my job as elementary music teacher in the Rossford School District after an exciting and rewarding summer. When reading students' responses to the question, "What did you most enjoy about music last year?" on a first-day questionnaire, I was pleased to see the number of students responding, *"the computer."* A Computer-Assisted Music Program at New Horizons Academy for the Gifted that was also held this past summer elicited a similar response among students. In both of these very different educational contexts, computer usage has proved to be a strong motivational force in students' acquisition of music theory and skills.

Last year my husband and I purchased a TI-99/4 with some rather nebulous ideas about potential applications in the general music curriculum. Together we worked on the development of programs—trying to incorporate motivational strategies which apply in virtually any teaching situation—and experimented with various uses of commercially available software (e.g., *Music Maker Command Module* and *Music Skills Trainer*). We tried many techniques in the classroom to determine the children's responses. The result has been the continuing evolution of a computer-assisted music curriculum tailored to the needs of my teaching situation. It has been a stimulating year—one which progressed from ignorance about using the computer and apprehensions concerning its effectiveness, to one of the most exciting experiences of my teaching career.

Whether you are a teacher planning a CAI project or a parent considering the potential educational value of a computer in your home, it may not be necessary to know exactly where you are going before you take the first step. Our experience has shown that the element of discovery inherent in developing a curriculum interactively with children can be as rewarding and exciting a learning process for the educator as is the use of the final product for the student.

**"... it may not be necessary to know exactly where you are going before you take the first step. ... discovery ... interactively with children can be as rewarding and exciting a learning process for the educator as is the use of the final product for the student."**

Glenwood School in the Rossford (Ohio) District is a typical elementary school with an enrollment of four hundred students in grades 1-6. I incorporated the computer in my general music curriculum for grades 4-6 during the fall semester of the 1980-81 school year (my first year in this system). Classes were intact groups which met for two thirty-five minute periods each week in the "music room"—a corner of the cafeteria. Average class size was twenty-five students with pupils from the Adjusted Curriculum (learning difficulties) as well as Project Horizons (gifted) mainstreamed into the regular classes.

My classes are organized around the belief that music should be fun and provide students with an outlet for their creativity. Although music class can be a break from routine academics, children must be equipped with a basic knowledge of the fundamentals of music reading and theory upon completion of a general music course. A variety of experiences should be provided such as singing, movement, listening, playing instruments, etc. The computer was employed as an additional enrichment activity—one which turned out to be unusually effective for the students, as well as challenging for the teacher.

The general approach I employ involves an initial experiential emphasis (e.g., singing and movement) followed by instruction in the basic elements of theory required to read music. In addition to providing knowledge of theory, this approach readies students for potential participation in band and choir.

Two computer programs were used, "Rhythm" and "Mystery Words," to reinforce two aspects of the curriculum: (1) aural recognition of rhythmic patterns, and (2) knowledge of musical notation for note names in both treble and bass clef.

In teaching students how to discriminate between various note values and rests, I first used "Echo Clapping" in which I clapped a rhythmic pattern and the students tried to reproduce it. Next, students were taught to associate appropriate music terms with relative dura-

**About the Authors**
*Mrs. Clulow has had 12 years experience as an instrumental, vocal, and general music teacher in grades 1-12. She received a B.S. in Music Education from Ohio State University and an M.Ed. in Elementary Education from Bowling Green University. Mr. Clulow received an M.Ed. degree in Psychoeducational Processes from Temple University and has completed all coursework and research requirements in the PhD program. He is employed as a chemist.*

tions (whole, half, quarter, eighth, and their corresponding rests).

Then we progressed to an activity called "Rhythmic Dictation" in which students wrote in musical notation the rhythms they heard clapped (e.g., ♩♩♩♩, ♩𝄽♩𝄽, ♩♩♫♩). So as not to get too complex at the beginning, only the ♩, ♫, and 𝄽 were taught.

Following presentation of the concepts and introduction to initial rhythmic dictation exercises, many students tend to reach a plateau in acquiring the skill of describing rhythmic patterns in musical notation. Representation of rhythmic patterns by clapping is to some extent an abstraction because it requires analysis of the relationship be tween a steady beat and intervals between claps. Although most children are highly motivated to acquire this skill, many encounter difficulties which result, in part, from its abstract nature.

To address this problem, we tried to incorporate two principles in the "Rhythm" program: (1) concreteness, and (2) immediacy of feedback. At the same time, in order to optimize the effect of one computer on twenty-five children, we decided to use a game format—allowing for the possibility of up to ten teams within a class. We also put a lot of effort into the introductory portion of the program to catch students' attention with the color and sound capabilities of the TI-99/4.

I initially introduced this program in my sixth grade classes after considerable time had been spent practicing rhythmic dictation. Periodic quizzes showed that a fair number of students in all three sixth grade classes had not grasped the concept. After three sessions with the "Rhythm" program, almost every student had become competent in clapped rhythmic dictation.

I believe several factors contributed to this remarkable improvement: The activity was conceptually more concrete than rhythmic dictation, and it provided students with immediate feedback which included the correct response when mistakes were made. Concentration and motivation were improved, in part, simply by the uniqueness of the computer activity. It was not uncommon for several students to show up in the music room shortly after their bus arrived in the hope of spending ten minutes working with a computer music game before school started. When teachers arrived after a general music class to take their students back to their classrooms, we invariably had to pry some of them away; classes sometimes had to be actually extended ten to fifteen minutes!

The group dynamics involved also played a significant role. During the first session, the sixth graders asked to have team scores added together for comparison with the other sixth grade classes. Because each class represented an intact group with some history and cohesiveness, a positive atmosphere prevailed in which the student working at the computer was supported and encouraged by the cheers and comments of fellow classmates. All students had several opportunities to make a contribution to the class total score.

Next, I used the "Rhythm" program in the fifth grade. Most students were aware through word-of-mouth that the sixth graders had been using a computer, and naturally they were interested in the top score the sixth graders had achieved. The typical score for the first day in the sixth grade had been 15 to 20. By comparison, the first day scores in the fifth grade were as high as 45! Similar enthusiasm was observed in the fourth grade.

For the most part, I anticipated these outcomes—although the actual impressive results far exceeded my expectations. There were also some genuine surprises: First, using the computer allowed me to observe and diagnose the

**"Concentration and motivation were improved, in part, simply by the uniqueness of the computer activity."**

problems of individual students and, where necessary, to take them aside and give special attention to their needs while the class was occupied with the computer. Learning the concept was important to them in order to make points for their class.

Another gratifying result was that the Adjusted Curriculum students found it easier to master this more concrete activity and took a great deal of pride in their contributions to the class score. It was indeed rewarding to see their beaming faces as classmates cheered and patted them on the back after their correct responses.

Following the computer activity, nearly every student had achieved competency in the basic rhythms which had been presented, and they were able to apply this knowledge in the playing of rhythm instruments. I wrote several lines of rhythmic patterns on the board in musical notation and asked individuals or small groups to play them. Subsequently, the patterns were played to accompany class singing or listening to records.

The rhythm unit was followed by the study of musical notation for pitch. Students were introduced to this concept through a discussion of the importance of learning the note names on the staff in order to read music when singing or playing instruments. I compared note reading to reading a foreign language —with symbols and notes used instead of words, creating a musical story.

Initial instruction presented the familiar "Every Good Boy Does Fine" and "F A C E" to facilitate learning the position of notes in the treble clef, and this was followed by drills to further reinforce note name recognition. Students were promised that the computer would be brought back to class when they learned these note names well enough to play a computer game. Thereafter, the "Mystery Words" game listed at the end of this article was introduced.

"Mystery Words" is a game that is based upon the fact that note name letters can be used to spell a variety of words, for example, "cabbage," "bead," "facade," etc. The program randomly chooses one of these words and represents it in music notation graphics in the treble clef, the bass clef, or both. The teacher has the option of excluding words with more difficult meanings (such as "facade" or "accede") when using the game with younger students.

The screen is divided in half with a red side and a blue side corresponding to the red and blue teams into which the class is divided. One member of each team is seated at the console. Before the presentation of a Mystery Word each player must signal he is ready by pressing the "1" or "0" key. As each team member signals readiness, a traffic light on each side of the screen changes from red to yellow to green, the Mystery Theme is played, and the graphic representation of the Mystery Word appears. The first student to decipher the word presses the 1 or 0 key, and the graphic representation disappears. He is then instructed to enter the answer using the keys 3 through 9 which have been labeled A through G on a blank keyboard overlay. As each letter is pressed, it appears in the graphics window. If the entire word is entered correctly, the graphic representation reappears with notes above the letters entered by the student, and the team's score is incremented. In the event an incorrect letter is entered, the opposing team member is instructed to try.

When the game was introduced in class, only the treble clef option was selected since previous instruction did not include the bass clef. Prior experience suggested that the presentation of both treble and bass clefs was too confusing for the average elementary age student. However, using the computer, treble clef note names were quickly mastered and the students requested that they be allowed to try working with the bass clef as well. Their ability to rapidly learn bass clef note names with minimal prior classroom work and to work with both clefs simultaneously was truly amazing.

Use of "Mystery Words" was accompanied by the same intense interest and motivation as "Rhythm." Although this game was also constructed for intra-class competition, students again asked that team scores be added together for comparison with other classes. Some students began showing up before school with younger brothers and sisters to explain the computer games to them, and I became a popular figure among students in the cafeteria at lunch time—the main topic of conversation being the computer.

Three components of an "appropriable" mathematics recommended by Papert (1980) are the *Continuity Principle*, the *Power Principle*, and the *Principle of Cultural Resonance*. These principles, of course, may be applied to the acquisition of *any* content domain—not just mathematics. This is to say that a concept or skill may be acquired with the least effort if it is (1) continuous with what the learner already knows, (2) if it empowers him to achieve personal objectives which could not be achieved otherwise, and (3) if it makes sense within a larger social context. Construing the computer-assisted units on these principles may help to elucidate some of the elements which I believe are critical to the success of this application (and for that matter, of any learning environment).

All children are intimately familiar with music in their everyday environment. The initial emphasis on those aspects of music already familiar to them, i.e., singing and movement, helps to give a sense of continuity with respect to subsequent course material. Second, the computer activity was integrated into a larger social context by the students themselves when they asked that team scores be added together for comparison with other classes. This phenomenon was also apparent when students in lower grades expressed interest in the scores obtained by upper grades, and their use of this information in the setting of personal goals.

But perhaps the most important element is the *Power Principle*. Students perceived that the acquisition of music skills would enable them to make contributions toward achievement of the group goal in the computer music game. Later they found that they were able to play rhythm instruments and read musical notation, and at the same time, they realized that these skills may be useful to them in the future with respect to participation in band and chorus—activities which are themselves part of a meaningful social context.

In summary, use of the computer increased student motivation, and at the same time allowed abstract material to be represented in a concrete way, leading to more rapid acquisition of skills and concepts. The computer also made it possible to diagnose individual weaknesses and provide individualized remedial work. Discipline problems arising from boredom and lack of interest were non-existent when the computer was in the classroom. In general, a positive environment, cooperation, and mutual support predominated.

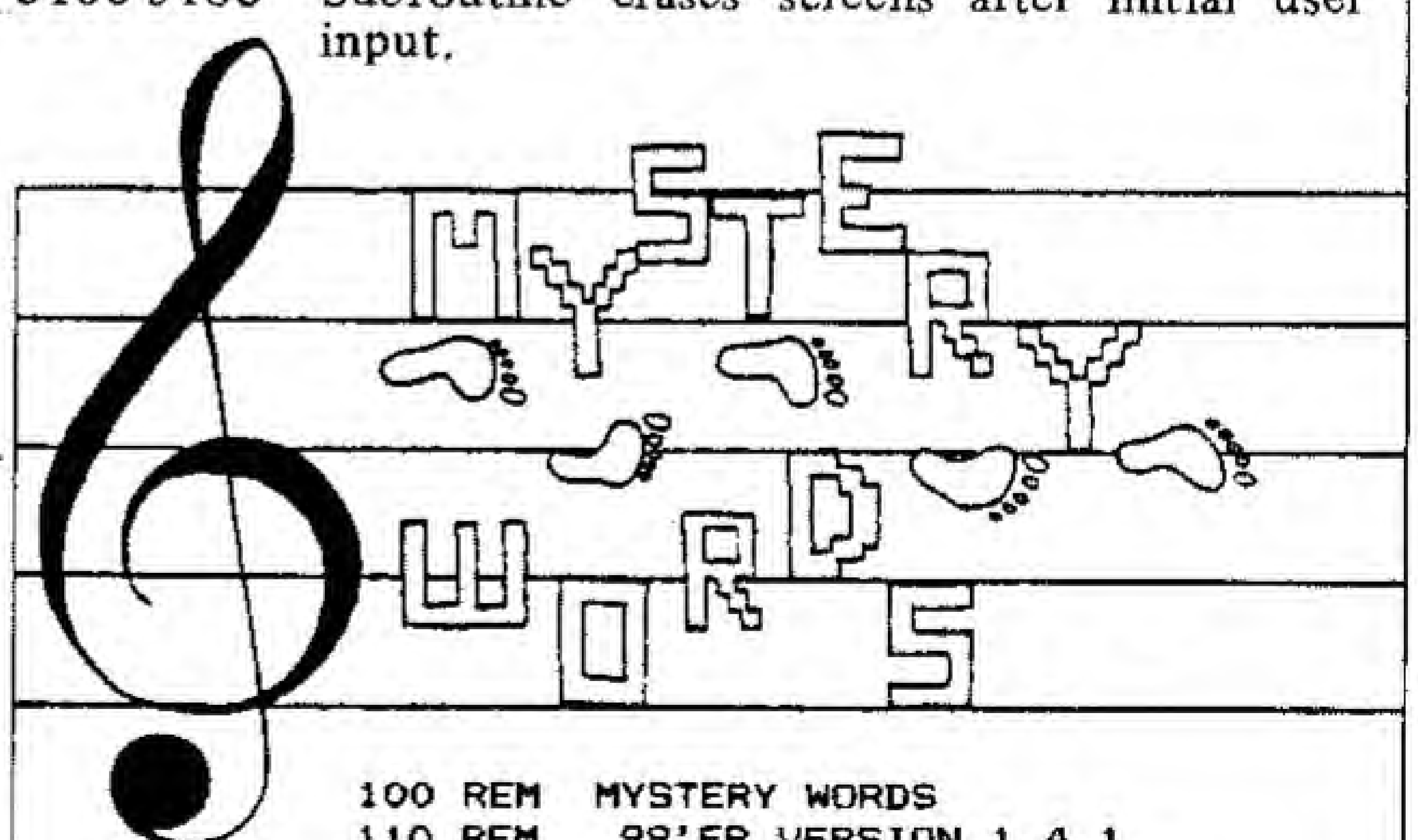
Part II of this series will describe a computer music experience in a very different educational environment—a summer workshop for gifted children at New Horizons Academy, in which students 7 to 13 years of age learned to write their own music programs in BASIC.

**Reference**

Papert, S. *Mindstorms: Children, Computers, and Powerful Ideas.* New york: Basic Books, Inc., 1980.

EXPLANATION OF THE PROGRAM
*Mystery Notes*

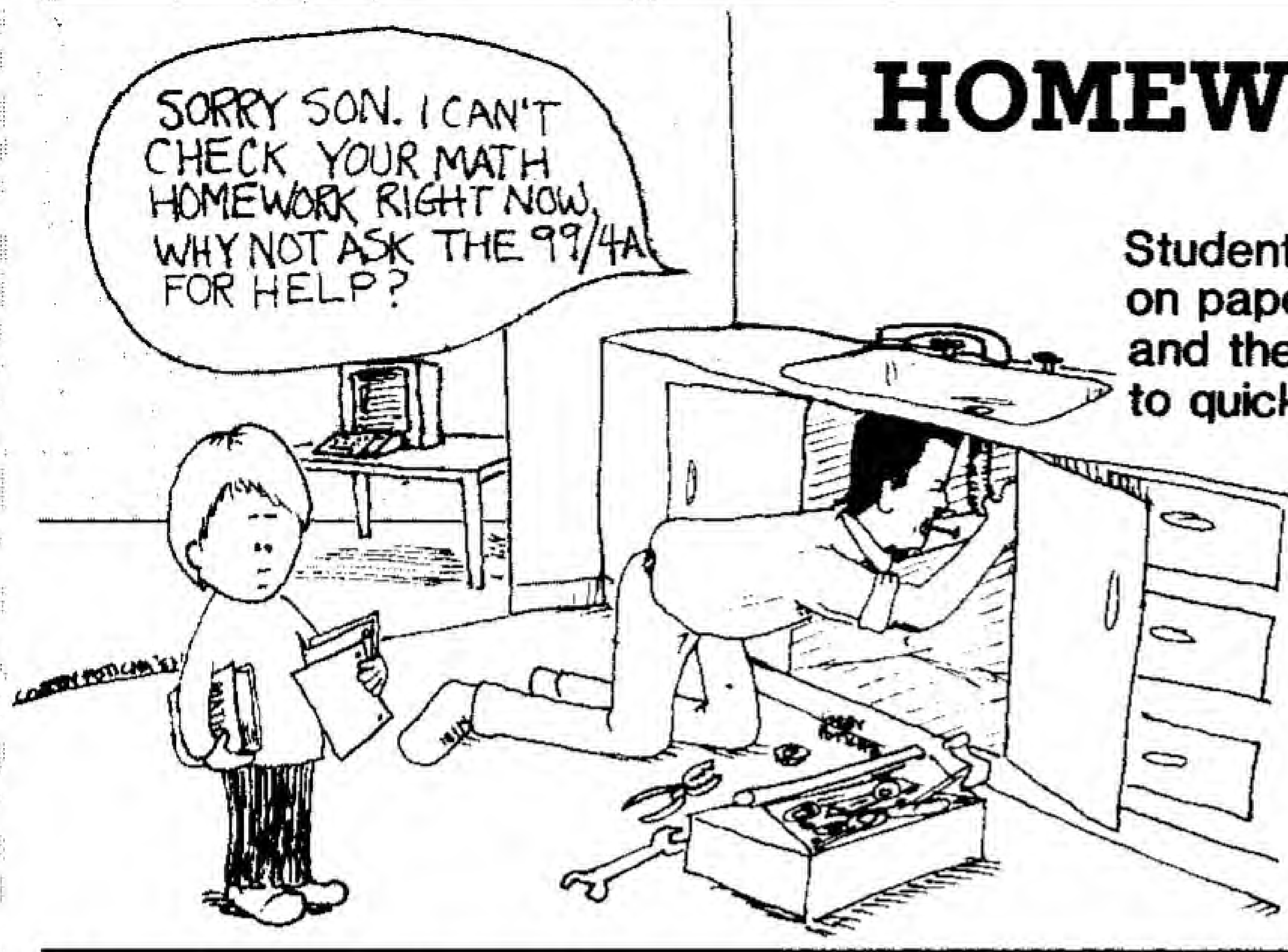
| Line Nos. | |
|---|---|
| 130-200 | Clears screen; sets screen color. |
| 210-270 | Accepts user input for word list. |
| 280-350 | Accepts user choice type of drill. |
| 360-420 | Instruction for keyboard labeling. |
| 430-490 | Interrupts program for explanation to class. |
| 500-700 | Displays notes of treble clef for review. |
| 710-840 | Displays notes of bass clef for review. |
| 850-870 | Restores data pointer and dimensions arrays. |
| 888-980 | Character patterns and set colors for initial graphic screen. |
| 990-1100 | Displays initial graphics screen; plays associated sounds. |
| 1110-1330 | Displays Mystery Words title screen. |
| 1340-1390 | Assigns word codes to SETS$ array; numeric digits correspond to letters (1=A, 2=B, etc.) |
| 1400-1470 | Assigns coded print locations (potentially 2 for each staff) for each of the 7 letter codes to array LINE. |
| 1480-1920 | Character patterns and colors used in the game screen. |
| 1930-2090 | Displays basic elements of game screen. |
| 2100-2110 | Selects a mystery word randomly. |
| 2120-2530 | Determines a representation for word on staff(s); when more than 1 possibility for a location exists, choice is random. The word is played in invisable characters (white on white) |
| 2540-3000 | Displays READY message, changes traffic light colors as players signal ready by pessing keys 1 and 0; plays mystery theme. |
| 3010-3030 | Makes representation of mystery word visable. |
| 3040-3100 | Accepts answer signals from players. |
| 3110-3180 | Flashes screen of red team. |
| 3190-3250 | Flashes screen of blue team. |
| 3260-3280 | Removes mystery word from view pending players response. |
| 3290-3300 | Instructs team to answer. |
| 3310-3450 | Accepts keys 3-9 (A-G) input, checking each letter against correct spelling. |
| 3460-3690 | If answer is correct, displays mystery word again with spelling; increments team score; returns to line 1930. |
| 3700-3820 | If answer is incorrect, allows opposing team to respond. |
| 3830-4330 | Subroutine displays game screen. |
| 4340-4470 | Subroutine draws treble staff. |
| 4480-4570 | Subroutine draws bass staff. |
| 4580-4630 | Subroutine displays messages, MSG$. |
| 4640-4700 | Subroutine displays letters of answer. |
| 4710-4830 | Subroutine displays initial graphic screen. |
| 4840-4900 | Subroutine plays mystery theme. |
| 4910-5140 | Subroutine displays treble and/or bass clef notes for initial review. |
| 5160-5180 | Subroutine erases screens after initial user input. |



```
100 REM  MYSTERY WORDS
110 REM   99'ER VERSION 1.4.1
120 REM  NORMA AND JOHN CLULOW
130 CALL CLEAR
140 NSET=52
150 RANDOMIZE
160 CALL SCREEN(5)
170 CALL VCHAR(1,31,1,96)
180 FOR I=1 TO 8
190 CALL COLOR(I,2,8)
200 NEXT I
```

Continued on p. 61

# HOMEWORK HELPER:


SORRY SON. I CAN'T CHECK YOUR MATH HOMEWORK RIGHT NOW, WHY NOT ASK THE 99/4A FOR HELP?

**Students do their class assignments on paper in the usual way . . . . and then can use the Homework Helper to quickly correct their assignments.**

# DIVISION

By Regena

The *Homework Helper* series of programs is designed to quickly give answers to students checking their homework assignments. *Division* gives the answers to three types of homework problems an elementary school student may encounter: division with a remainder, division with a decimal in the quotient, and division in fraction form of a numerator by the denominator.

Only the answers are given, *not* the step-by-step process of long division. The student is encouraged to do his or her homework—writing each step in the division process and then using this program to check the answers. Music and graphics enhance the interaction.

## 1. Division with Remainder

Most math problems can simply be corrected with a calculator. However, if there is a remainder, a calculator converts it to a decimal equivalent. This program keeps the answer in quotient-plus-remainder form. The student enters the divisor and dividend; the quotient and remainder are printed.

## 2. Division with Decimal

Usually after students master the idea of a remainder, they are taught how to place a decimal and keep dividing. In this section, a student enters the divisor and dividend; the quotient with a decimal fraction is printed.

## 3. Convert Fraction to Decimal

A fraction is converted to a decimal by dividing the numerator by the denominator. The student enters the numerator then the denominator; the equivalent decimal fraction is returned.

After each problem, a student may enter another problem of the same type. If there are no more problems of the same kind or the student wishes to end, he enters zero and the menu screen will return.

EXPLANATION OF THE PROGRAM
*Homework Helper—Division*

| Line Nos. | |
|---|---|
| 170-810 | Prints title screen and blinks colors while special graphics characters are defined. |
| 820-1720 | Plays music; prints menu screen and branches appropriately for student's response. |
| 1730-1850 | Subroutine to print labels of division problem. |
| 1860-1980 | Routine for division with remainder. |
| 1990-2090 | Routine for division with decimal. |
| 2100-2360 | Routine for converting fraction to decimal. |

```
100 REM  *******************
110 REM  * HOMEWORK HELPER *
120 REM  *    DIVISION     *
130 REM  *******************
140 REM  BY REGENA
150 REM   99'ER VERSION 1.4.1
160 REM
170 T=300
180 CALL CLEAR
190 CALL CHAR(96,"0000784444444478")
200 CALL CHAR(97,"0000381010101038")
210 CALL CHAR(98,"0000444444442810")
220 PRINT TAB(7);"H O M E W O R K"
230 CALL CHAR(99,"0000384430084438")
240 PRINT :TAB(9);"H E L P E R"
250 CALL CHAR(100,"00007C444444447C")
260 CALL CHAR(101,"0000784444784844")
270 PRINT ::::::TAB(7);"D I V I S I O N"::::::::
280 CALL CHAR(102,"3F10100808l0102")
290 CALL CHAR(103,"FF00784444444478")
300 CALL COLOR(2,7,16)
310 CALL HCHAR(14,7,42,19)
320 CALL VCHAR(15,7,42,3)
330 CALL VCHAR(15,25,42,3)
340 CALL HCHAR(18,7,42,19)
350 CALL COLOR(2,16,7)
360 CALL CHAR(104,"FF00381010101038")
370 CALL CHAR(105,"FF0044444444281")
380 CALL COLOR(2,7,16)
390 CALL CHAR(106,"FF007C407840407C")
400 CALL CHAR(107,"FF00446454544C44")
410 CALL COLOR(2,16,7)
420 CALL CHAR(108,"0172FCFEFEFE7E3C")
430 CALL CHAR(112,"384444544C3C")
440 CALL COLOR(2,7,16)
450 CALL CHAR(113,"444444444438")
460 CALL CHAR(114,"7C444444447C")
470 CALL COLOR(2,16,7)
460 CALL CHAR(115,"7C101010101")
490 CALL CHAR(116,"381010101038")
500 CALL COLOR(2,7,16)
510 CALL CHAR(117,"7C407840407C")
520 CALL CHAR(118,"446454544C44")
530 CALL COLOR(2,16,7)
540 CALL CHAR(119,"0078444478 4844")
550 CALL CHAR(109,"0000040C1C3870E")
560 CALL COLOR(2,7,16)
570 CALL CHAR(110,"0103070E1C387")
580 CALL CHAR(111,"80C0E070381C0E07")
590 CALL COLOR(2,16,7)
600 CALL CHAR(136,"80C0E0F0F8FCFEFF")
610 CALL CHAR(137,"7F3F1F0F070301")
620 CALL COLOR(2,7,16)
630 CALL CHAR(121,"000000003070F0F")
640 CALL COLOR(2,16,7)
650 CALL CHAR(122,"030F3FFFFFFFFFFF")
660 CALL CHAR(123,"C0F0FCFFFFFFFFFF")
670 CALL COLOR(2,7,16)
680 CALL CHAR(124,"00000000C0E0F0F")
690 CALL CHAR(125,"1F1F3F3F7F7FFFFF")
```

Continued on p. 83

# Computer Techniques for Tutoring the MENTALLY Handicapped

By Sam L. Jenkins and Jerry Kirsling
*University of Wisconsin-Stout*
*Stout Vocational Rehabilitation Institute*
*Menomonie, WI 54751*

Alas, the revolution has just started! And the fact that you're reading *99'er Magazine* signifies that you are very much a part of it—a revolution fueled by the availability and affordability of computer power to millions of consumers. As more and more software is developed—computer programs that can meet a large diversity of everyday needs, as well as solve problems encountered in special areas—the computer will become as common in our homes as the telephone.

Our task in this generation is to learn to take advantage of this tool in a variety of areas, disciplines, and endeavors. In this article, we would like to focus the application of computer technology on what may seem at first to be a most unlikely area—tutoring the developmentally disabled.

Retardation refers to "below average intellectual functioning that originates during the developmental stages with associated maladaptive behavior." In the search for tools to combat retardation, the microcomputer has shown itself to be extremely valuable in assisting the retarded population in developing skills, abilities, concepts, and even behaviors. Preliminary testing demonstrates that not only can these individuals use a keyboard, but they can learn it very quickly—finding it attractive, novel, and magnetic. Such options as the light pen, joystick, and voice synthesizer provide capabilities that can be used to adapt numerous programs for this special population.

## Help for the Schools

For more than eight million handicapped children in the U.S., only half are reportedly receiving appropriate educational services. School districts under ever-tightening budgets struggle to meet the needs of these children. It therefore appears highly probable that using microcomputers to assist in meeting the needs of these children will be both an economic boon to schools, and a valuable enhancement to the learning process of these youngsters.

Apart from traditional controversies regarding the learning process, there are some areas of general agreement. These areas have provided us with a basis for a type of software geared to the special learning needs of the retarded—a program utilizing the unique qualities of the computer to further stimulate learning.

## Fascination With the Medium

Retarded and non-retarded alike are able to learn more and easier from teaching aids that effectively focus their attention on the content. Attention management for the retarded youngster is especially critical. In this regard, the computer, keyboard, and CRT have a fascination that commands attention with an immediacy that is unparalleled. When seating a youngster before a console, the attraction of the mechanism coupled with the allure of a good interactive program provide an incredible amount of motivation and drive. If you have youngsters who play computer games or other electronic games using a microprocessor, you already know just how easy it is to distract them and draw their attention to something else like homework, eating, or cleaning their room!

"... the computer, keyboard, and CRT have a fascination that commands attention with an immediacy that is unparalled."

## Nothing Succeeds Like Success

Another factor operating here is the principle that as human beings we tend to strive toward success or try to avoid failure. In the search for success, the "locus of control" is usually internal. This is to say that in the process of maturing, a person begins to realize a power or ability to control what happens, and begins to set goals. An efficiency in goal attainment begins to occur and a sense of success starts to emerge. Attaining goals brings a sense of success which is its own reward, and prompts one to continue to strive for success.

Avoiding failure, on the other hand, is to maintain a mere minimum of effort so as not to incur some type of punishment. Consequently, the locus of control is *external*. For a majority of the developmentally disabled, avoiding punishment becomes the usual way of behaving. They are not given to setting goals since they have not come to experience the internal locus of control and the possiblity of success.

With the use of computers, a learning environment can be created which can provide a retarded child or adult with the experience of success. As this experience is repeated, the locus of control begins to shift from without to within. This is a natural reward process which has more lasting effects than punishment or negative reinforcement. As the repertoire is gradually expanded, the retarded individual begins to realize a potential—a power for success.

## A Multisensory Lens

Another important element in the learning process of the retarded person is the ability to focus in on significant cues. Once again, the hardware's attractiveness (or novelty, if you will) is so engaging and attention-riveting (thereby limiting external or irrelevant stimuli or signals) that the person learns to be attentive to only the important and discriminating cues. Furthermore, the multisensory impact of the computer provides an additional quality which is extremely valuable in the learning process of the retarded person. The more you can use, engage, and impact as many sensory modalities—and do it *repeatedly* in an interesting manner—the greater the likelihood of retention and learning.

## An Example Program

The following is a simple program designed for teaching retarded persons the extremely abstract concepts of number recognition, counting, and subtraction. We feel that the program demonstrates the principles stated in this article, as well as the uniqueness of the computer as an especially well suited tool to meeting the learning needs of the developmentally disabled. We wish to emphasize that the computer does *not* totally substitute for a teacher. The retarded individuals on whom we tested the program required personal assistance and encouragement at the beginning of the lesson. Reaction to the computer ranged from reluctance to fascination to eager enthusiasm. In some cases, we first used another program (a keyboard trainer) to familiarize the student with the key locations on the console. The TI-99/4 keyboard is highly suited for use by those unfamiliar with typewriters. We found it helpful, however, to cover the letter keys with masking tape to reduce distractions. Also, we noted some confusion created by the shift characters above each number—a small problem we hope to overcome by trying a number of key covers. Based on field testing of this program, we are convinced that this approach can be extended to many areas of work with this group—a group whose needs are so unique that conventional methods have been only moderately successful. Using this technology, there exists a potential for far greater success, and the possibility of doing things that were unthought of for this segment of the population.

## The Program

The program opens with several options which must be selected. First, the instructor is informed that a performance rating of the student's progress is available by pressing the AID key. This gives the number of trials, correct answers, and percent correct. Also, if you wish to reset the options, simply press the BACK command and re-enter. The AID and the BACK commands can be entered during the main lesson, thus giving the instructor flexibility in designing the most appropriate set of options suitable to the student's level of ability. The program has a speech selection option that permits using it without the Speech Synthesizer and Speech Editor Command Module. [The extensive use of graphics in this program precludes the use of the speech editor resident in the Extended Basic Command Module with its fewer character sets available—Ed.] Although the actual lesson is designed for non-readers, the initial option selection must be performed by an instructor or someone who can read. These options can be selected in any combination from the following list:

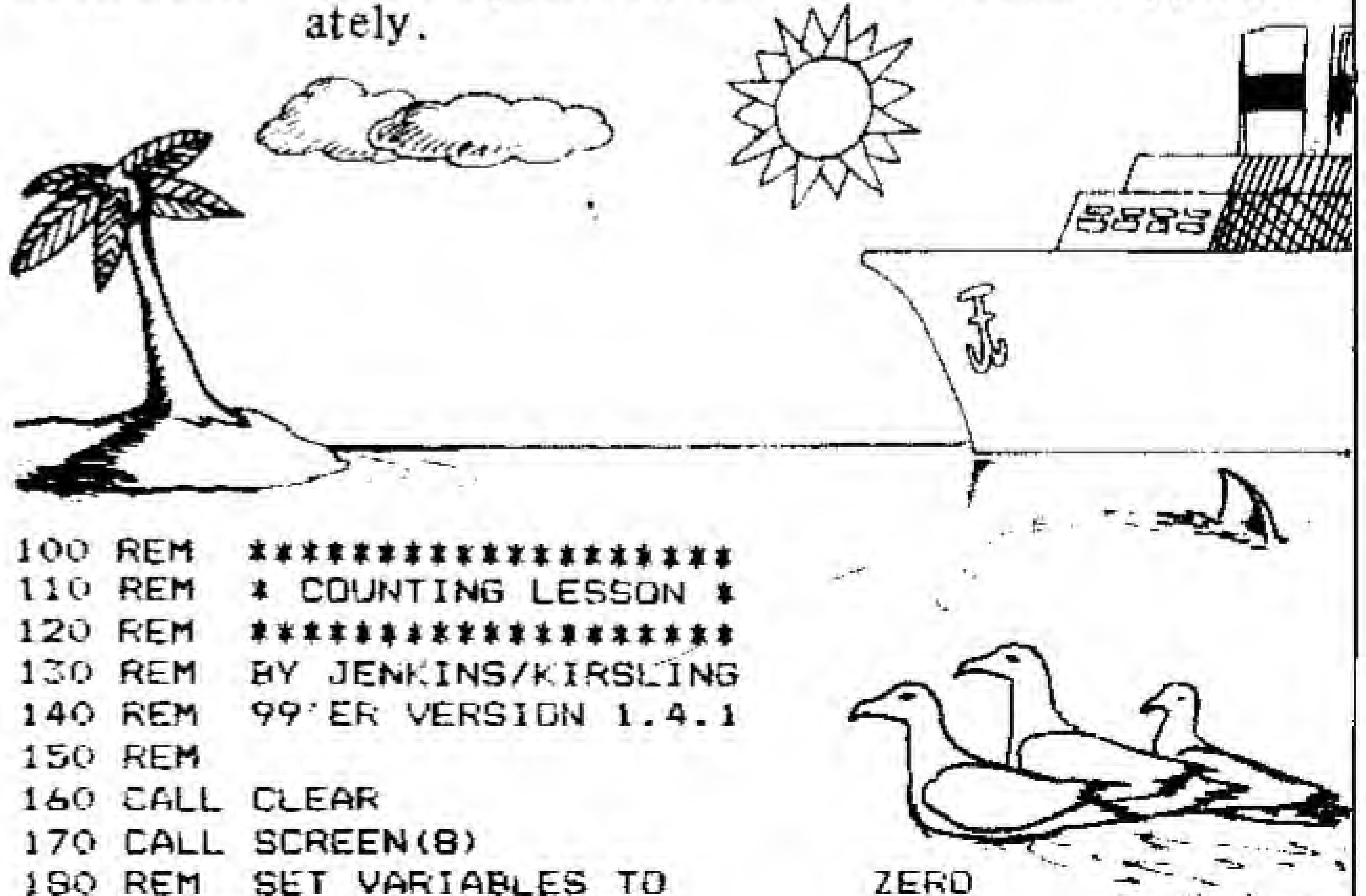
Select
1=RANDOM
2=Serial presentation
3=End lesson
—Display the number above the gulls (Y/N)
—Pronounce each number as it is printed (Y/N)
—Computer says press _ (number) after a row of gulls is put on the screen (Y/N)
**—Select format for placing gulls on the ocean**
1=Horizontal Row
2=Diagonal pattern
3=Random row placement

After the options are selected, the screen clears and a seascape is painted on the screen. A picture of a deep blue ocean and a steamship liner on the horizon moving toward a tropical island focusses the student's attention immediately. Depending on the options selected, the gulls appear on the water from left to right and a shark's fin begins to circle the gulls while waiting for the sutdent to press the key representing the number of gulls. If the response is correct, a musical fanfare is played followed by the computer saying "Right (number)," and the ship moves one column to the left, emiting smoke puffs from the stacks (the number of puffs equal to the number of gulls). However, if the student's response is incorrect, the computer says "Uhoh" and the shark stops circling the gulls, emerges from the water and devours the last gull (with sound effects)! Then the computer asks the student "What number is left?" and waits for the student to press the key representing the number of remaining gulls. If incorrect again, the computer says "That is incorrect" and gives a short laugh, then engulfs the next gull! This can continue until no gulls remain; the program then recycles and another trial begins. On a correct response the computer says "Right (number)" and the ship is advanced one column to the left with the appropriate number of smoke puffs. Each correct response advances the ship toward the island until it is "docked" and the computer says "You win." It then recycles the program, placing the ship back at the right side of the screen, and continues the lesson.

We recommend that students start with the serial option rather than the random. This starts with the number 1 and adds a number on each correct trial, but will not add a number on an error. In this way, a student cannot be challenged by the larger numbers until he has displayed mastery of the smaller ones. In general, we also recommend the strategy of starting a student with all prompt options operating, then removing them as the student demonstrates competence.

EXPLANATION OF THE PROGRAM
*Computer Techniques for Tutoring the Mentally Handicapped*

| Line Nos. | |
|---|---|
| 160-280 | Sets all variables to zero. |
| 290-820 | Instructor selects program options. |
| 830-1310 | Defines characters and color codes. |
| 1320-1450 | Constructs seascape, boat, and island. |
| 1460-1550 | Calculates the appropriate number of gulls to place on screen. |
| 1560-1590 | Clears screen from row 10 to row 24. |
| 1600-1820 | Places gulls in the water. |
| 1830-1890 | Controls movement of shark fin from left to right. |
| 1900-1960 | Evaluates key response while shark circles gulls. |
| 1970-2120 | Musical fanfare on correct response. |
| 2130-2220 | Controls movement of shark fin from right to left in background. |
| 2230-2290 | Evaluates key response. |
| 2300-2530 | Controls animation of shark eating gulls. |
| 2540-2620 | Evaluates key response and clears screen to right of last gull after shark "eats". |
| 2630-2660 | Controls loop to eat next gull. |
| 2670-2740 | Verbal response to a correct key press; increments trials and rights. |
| 2750-2910 | Moves ship, controls puffs of smoke and sound effects from ship stacks. |
| 2920-2950 | Prompts to press a number if a letter was pressed. |
| 2960-3060 | Routine when boat reaches island. |
| 3070-3100 | Calculates performance scores. |
| 3110-3170 | Prints option to end and branches appropriately. |



```
100 REM  ******************
110 REM  * COUNTING LESSON *
120 REM  ******************
130 REM  BY JENKINS/KIRSLING
140 REM  99'ER VERSION 1.4.1
150 REM
160 CALL CLEAR
170 CALL SCREEN(8)
180 REM  SET VARIABLES TO        ZERO
190 FOR LOOP=1 TO 10
```

Continued on p. 83

Streamline your administrative work

# School Management Applications

MICROCOMPUTER SOFTWARE PACKAGES

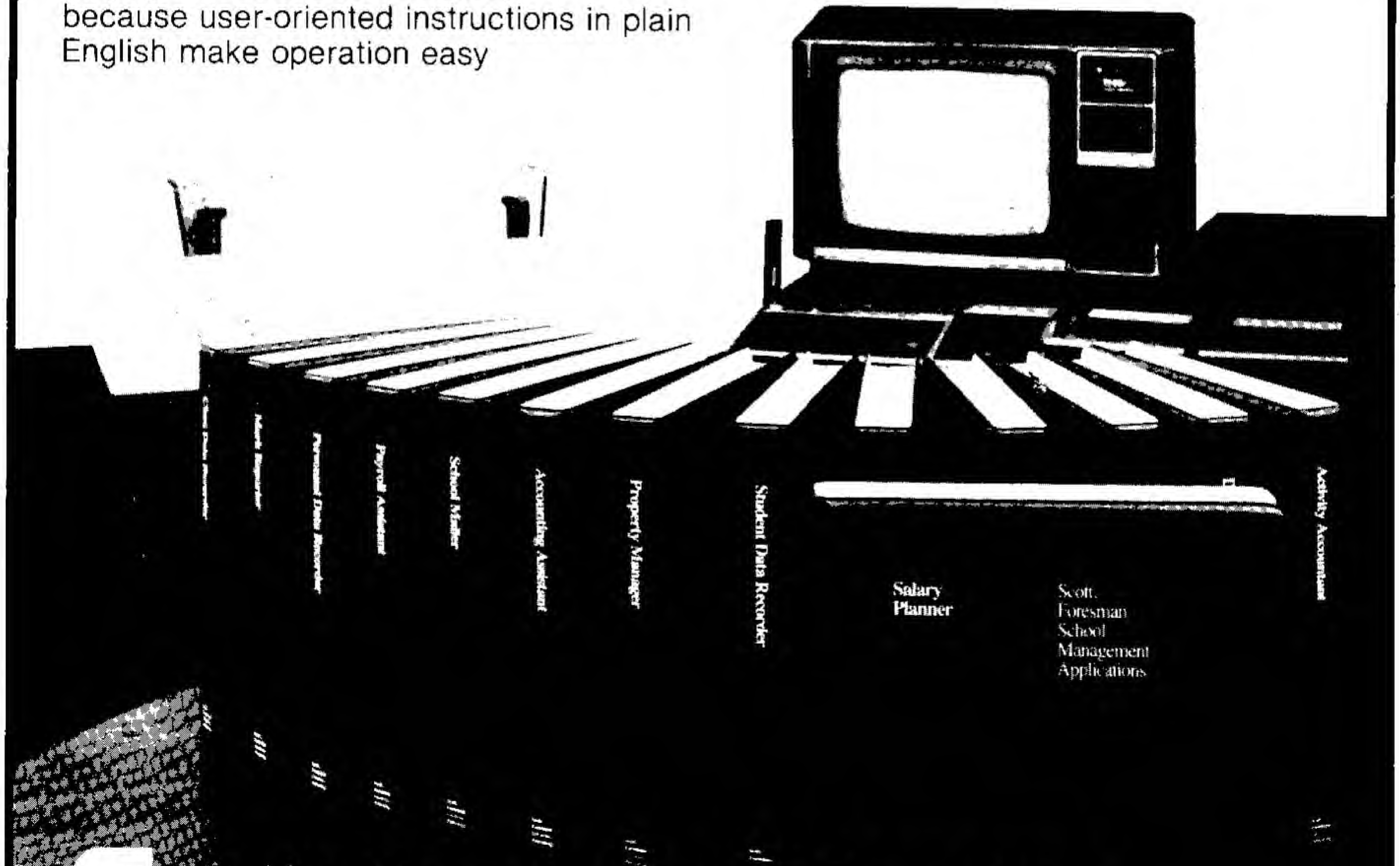
**Reduce paper work**
with professional, functional microcomputer programs

**Simplify record-keeping**
through convenient information storage capabilities

**Save time and money**
by producing reports, permanent records, mailing labels

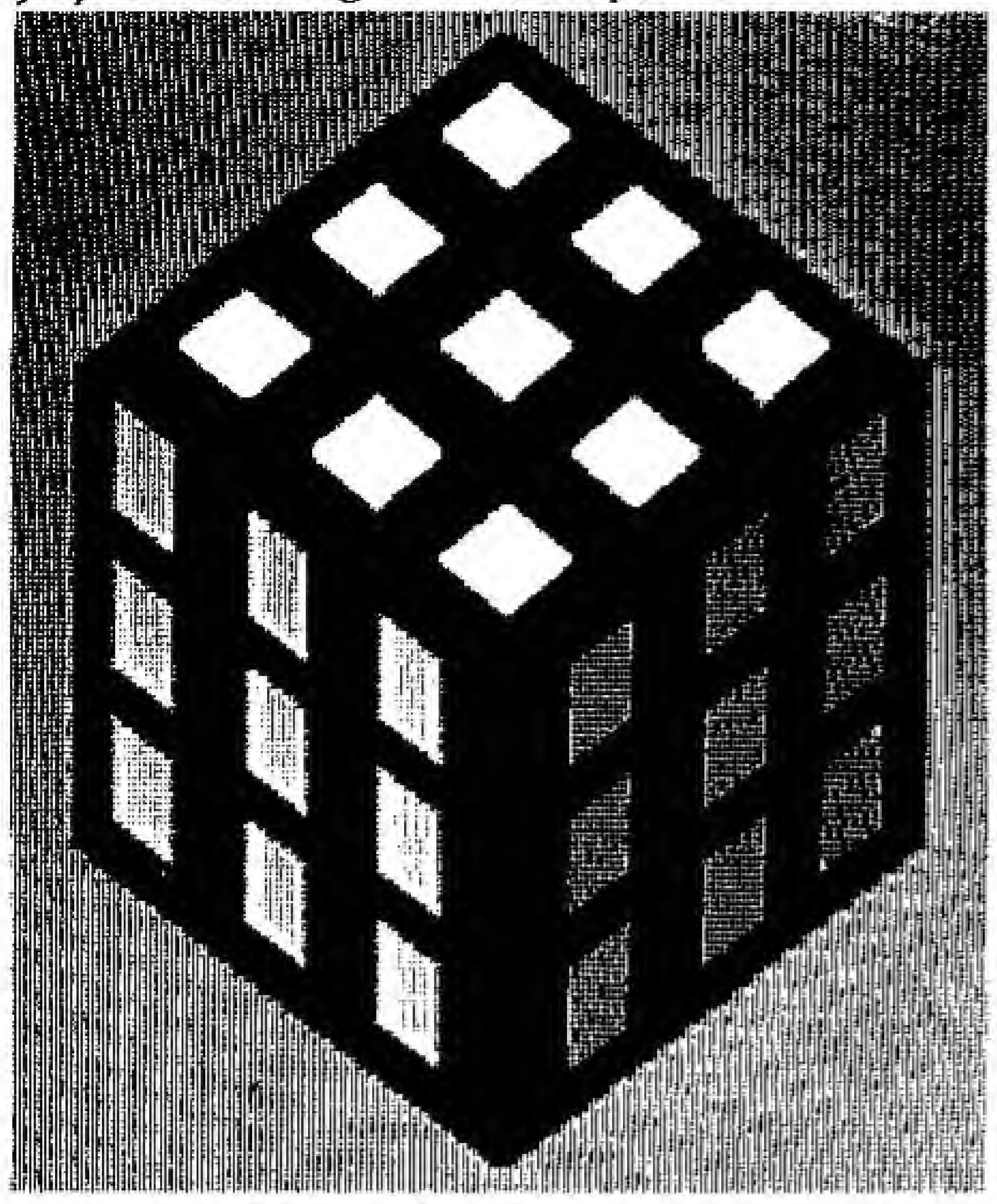
**Enjoy immediate results,**
because user-oriented instructions in plain English make operation easy

Each School Management Application helps you handle a specific area of your work—from class scheduling to test scoring to mark reporting. Learn more about them all. Send for free, full-color brochure 30661-5-NN2.



**Scott, Foresman and Company**
**Electronic Publishing**
**Glenview, Illinois 60025**

Joy of Gaming . . . from p. 12


Screen photo of The Cube from Linear Aesthetic Systems.

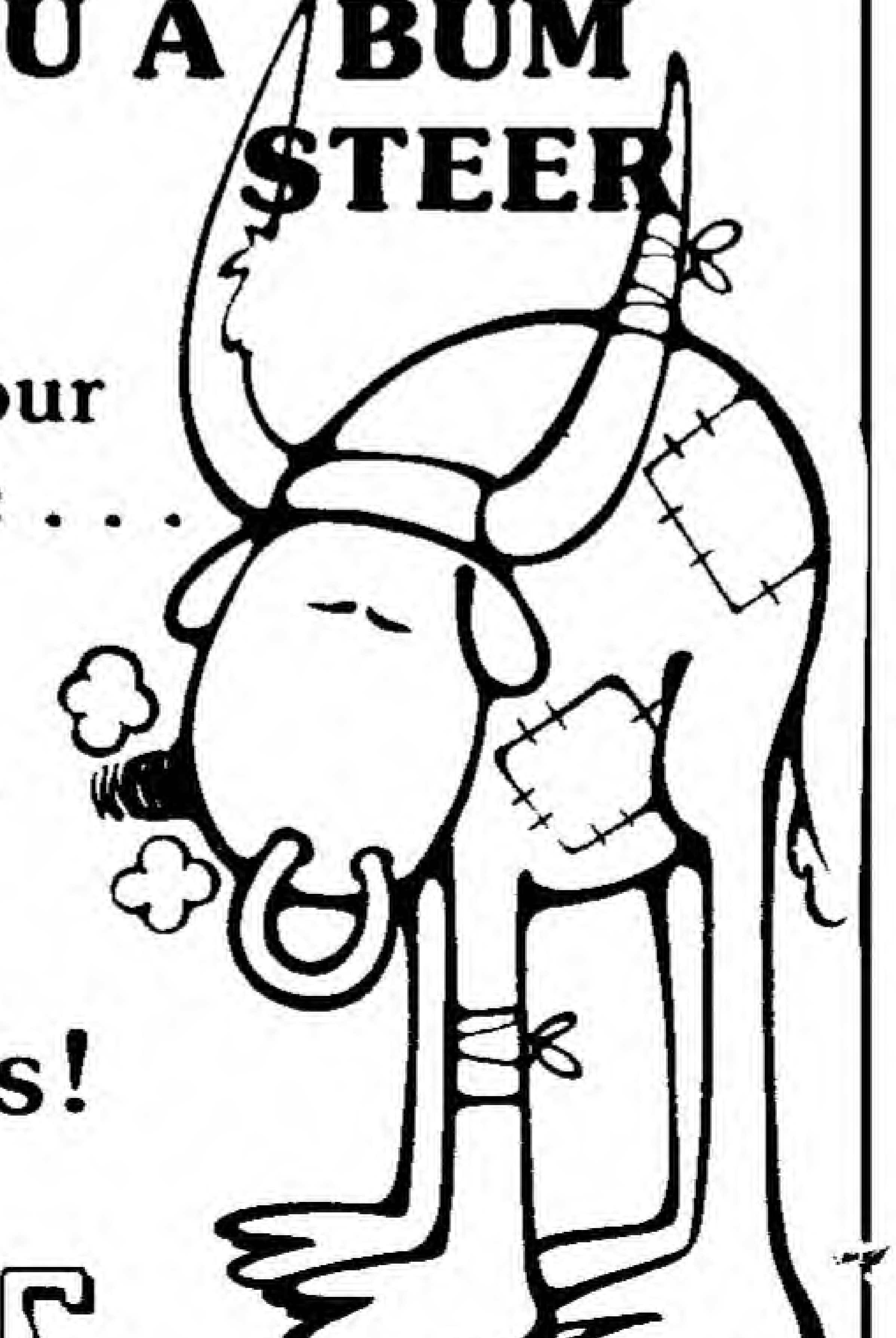
Since the relatively new capability of being able to create and control sprites is such a welcome addition to the programmer's arsenal, I've come to expect some form of "arcade effect" in all third-party Extended BASIC games. It therefore came as a complete surprise to find a pair of unique, thoroughly engrossing games that quite admirably took advantage of this language's enhancements *without* the obligatory, dazzling sprite movement. American Software Design & Distribution Co. (Cottage Grove, MN) has taken the basic raw elements of logical deduction, limited time, memory, and visual perception, then added a few measures of excitement and confusion, mixed' it all together with some interesting graphics, and has come up with *Bomb Squad* and *3-D Maze*. The first is a logical simulation that will have you on the edge of your seat while you ever-so-carefully test the wires and try to disarm a loudly ticking time bomb; the second is a spacially disorienting tour taken at eye-level, with the appearance of hallways and passages realistically changing in proper perspective according to how far you walk and the direction you turn.

And finally, rounding out my marathon of computer gaming, I came upon the popular card and word games that are usually programmed in TI BASIC. It was satisfying seeing the healthy diversity of these games. Some of the more novel titles among this batch were *Cribbage* from Sunshine Software (Hoffman Estates, IL), *American Baccarat* from Anthistle Systems (Ontario, Canada), and *Scribble* from PRP Computergraphics (Lake Charles, LA)—a two-player implementation of the popular *Scrabble* brand board game (with the computer monitoring the action and keeping score) that aficionados of this American classic should enjoy having available on their personal computers.

**DON'T LET ANYONE GIVE YOU A BUM STEER**

**When it Comes to Marketing Your Games Software . . .**

**For Expert Advice Call the Professionals!**



99'er magazine™

**ATTENTION**

**Game Players: Watch**

For the Next Issue of an Expanded 99'er Magazine™ For the Debut of

Computer Gaming™

**A NEW**

Magazine-within-a Magazine Featuring:

- Arcade Arbiter Review
- Strategy Corner
- 99'er Hall of Fame
- Designer's Spotlight
- Pros on Programming
- The Adventure Registry
- Joystick Jockey Q & A

Plus Many More Surprises . . .

**HELP !!!**

We need reader-written reviews (400-500 words) of computer games software—either TI or third-party. You may submit reviews on any of the programs already mentioned in the previous article, or on any other TI-compatible games software product. If your review is chosen for publication, it will become the property of 99'er Magazine for copyright purposes, and you will receive a different third-party or TI game as compensation.

Join the Arcade Arbiters—Send in Your Reviews Today!

TI fest™ *is Coming...*

See page 92

# at the WINTER CONSUMER

You gotta know when to hold,
Hear what you're told,
Know when to choose a card,
Know that you've won . . .

You never count your orders
While standin' on the show floor,
There'll be time enough for countin'
When the dealin's done.

Liberally paraphrased from: *The Gambler* by Kenny Rogers

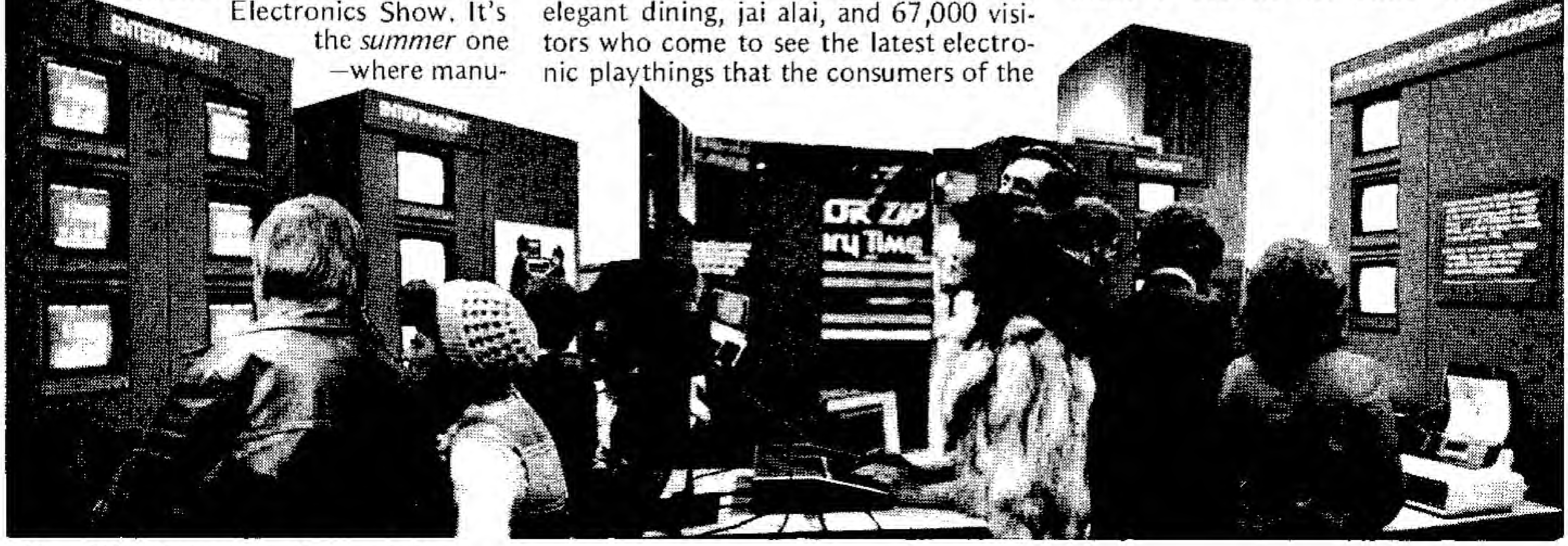
In early January, leaving the snow and ice behind, I flew off to Las Vegas. Gambling was definitely on my mind . . . Not the casino variety this time; but for an editor, something just as uncertain—the decision to "hold the front page," saving it for some really exciting news and photos that *might* soon be forthcoming from there. Not just "filler" or "boiler plate" mind you; but rather, a detailed report on significant new developments—a feature article that would more than justify both the publication delay and the setting aside of a frightening amount of empty space (otherwise known as a "gaping hole").

It was all the more a gamble because nothing much exciting typically happens at the *winter* Consumer Electronics Show. It's the *summer* one—where manufacturers' new product lines usually make their debut—that has the press running around, snapping pictures, and dashing off stories of the new electronic goodies that will be in the stores during the forthcoming Christmas buying season. Well, I knew the odds . . . but nevertheless, there was this gut feeling . . .

As usual, Lubbock (headquarters of TI's Consumer Products Group) wasn't talking. But this time around, the proverbial silence was *unusually* "deafening." Something BIG just had to be in the works . . . or was it just my overactive reporter's imagination?

Las Vegas in January is like any other place in the desert—that is, any other place with 55,000 first-class hotel rooms, big casinos, elaborate stage shows featuring America's top(less) entertainers, elegant dining, jai alai, and 67,000 visitors who come to see the latest electronic playthings that the consumers of the world will soon be consuming with voracious billion-dollar appetites. The town was as I had left it six weeks earlier at a business computer trade show—wonderously alive and teeming with excited visitors. Only this time, the air was noticeably crisper, and the visitors numbered *three times* as many. So, when I arrived at the Convention Center complex for the opening ceremonies, it was obvious from the early crowds that this *winter* CES would set a new attendance record—by surpassing even last summer's popular Chicago CES (see *99'er Magazine*, July/August 1981).

Jostling with the crowd of other early arrivees, I finally made my way inside the doors. Then, quickly locating the Press Room, I picked up my badge and made a bee-line for Aisle 2200—the

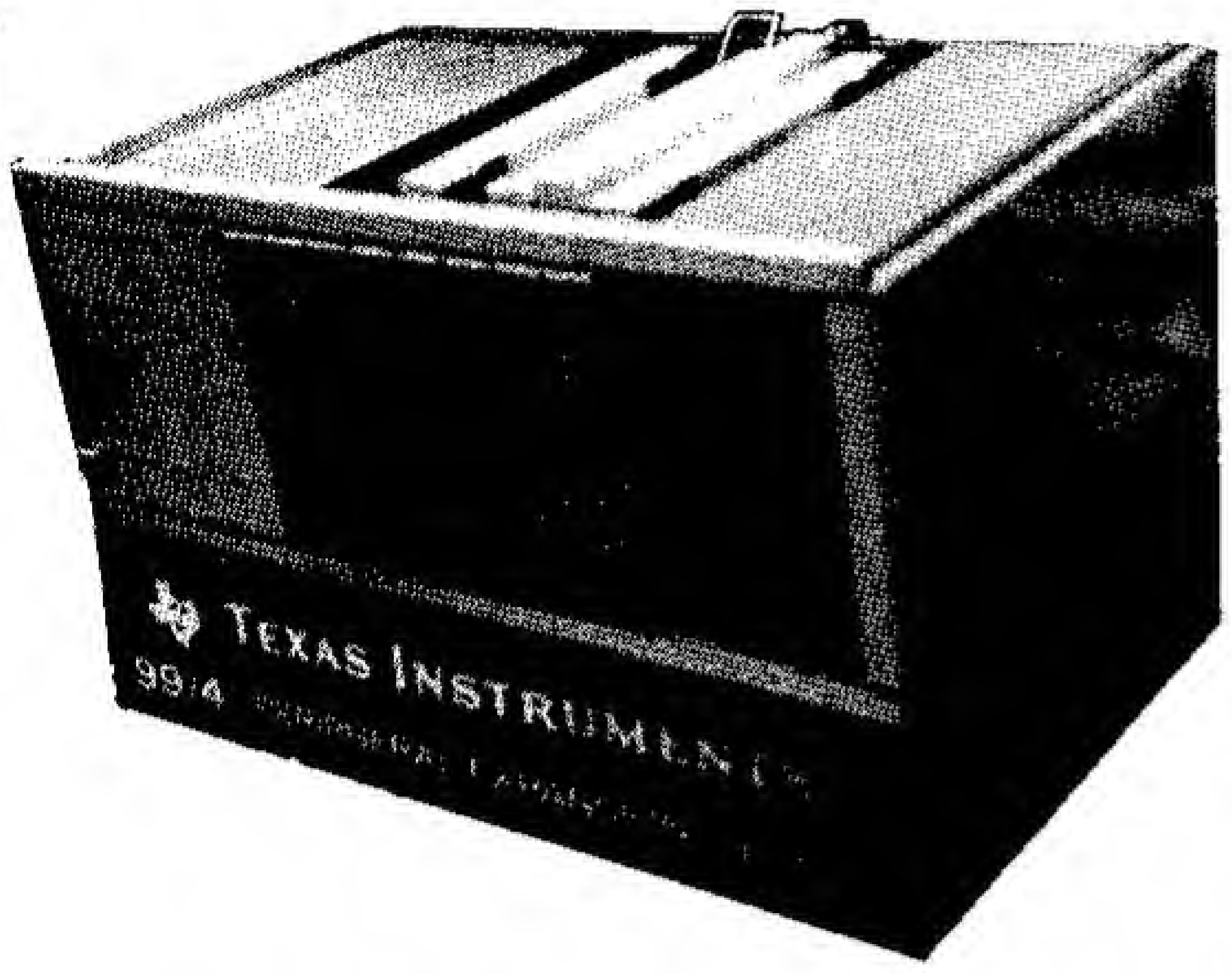


Vegas
7-10, 1982

# ELECTRONICS SHOW

By Gary M. Kaplan

• Introducing the new UCSD p-System* which enables the TI Home Computer to use new languages such as UCSD Pascal* . . . with FORTRAN, Compiled BASIC, and PILOT to become available in second quarter, 1982.
• Any user can now program TMS9900 Assembly language using either the UCSD p-System or the new stand-alone Editor/Assembler.
Texas Instruments



floor area where Texas Instruments would be waiting . . . As I walked through the huge building, plenty of sights and sounds vied for my attention—distractions that could have detoured a less-dedicated editor than "yours truly." There were glittering arrays of shiny new audio-visual hardware, exploding screenfulls of exciting video game software, wandering R2D2 lookalikes chattering away at passersby, cute Japanese girls gyrating to the disco sounds emanating from the latest in portable stereo headware, plus a bevy of bunnies and pets (of the *Playboy* and *Penthouse* variety). All these distractions, yet on I went . . . Finally, I caught a glimpse of the TI display in the distance, and the big question that brought me here loomed large and menacing in my mind: Would there really be anything truly significant to report on?

Would there, indeed. As it turned out, TI did introduce some very exciting new products—more than even one most optimistic reporter had hoped for. But news never gets any fresher with age, so as they say in the world of entertainment, "Let's get on with the show!"

## TIing One On & Missing the Train

What would *you* call a rectangular enclosure that expands your Texas Instruments personal computer system by tying together all its peripherals in one neat package? Texas Instruments was faced with this same naming problem and (rather unimaginatively) came up with—what else but—"Peripheral Expansion Box." I predict, however, that it will affectionately become known as a "TI-Box" (pronounced and punned "tie box"). Regardless of name, the new addition is an impressive sight. It's the perfect match for the silver and black TI-99/4A console and color monitor. And when sitting directly behind the console with the monitor perched atop, it creates a very professional looking workstation appearance—complete with amber-colored flashing lights (just like in the movies . . . ) that monitor system activity and double as an aid in diagnosing problems.

The first thing you notice when you look down inside the box is how well it's built. Down the center there are 8 slots for peripheral, board-type modules —one of which is used to connect the system to the computer console. Although this plug-in arrangement has been used extensively in many so called "hobbyist" systems (such as S-100 bus configurations and the Apple II with its built-in "motherboard"), the way TI chose to implement it is so far superior to these predecessors that the system *can't* be regarded as "hobbyist" (with its derogatory connotation of being too technical or confusing to hook up and operate for the average consumer): Careful human engineering is evident in the heavy-duty rail tracks that guide the module cards into position for mating with the connectors. There's virtually no way a user can insert them backwards or snap off pins. The "cards" themselves are special—enclosed in metallic panels with no exposed components or wiring to short out, crimp, or otherwise damage. These peripheral modules shouldn't present any problem for consumers to quickly install (or pull out with the attached "puller rings"). Frankly, I wondered how TI could ever get away from their "freight-train" configuration—caused by the in-line, separate, plug-in peripherals while at the same time *maintaining* the configuration's fool-proof, peripheral "connectability." Now I know . . .

On the left side of the box is a heavy-duty power supply—complete with a cooling fan—that powers all the peripheral modules. TI evidenly engineered this box to provide years of trouble-free service—unlike some of the "motherboard" systems I've seen that are

characteristically plagued by insufficient power and cooling.

The right side of the box provides space for installing one TI disk drive mounted vertically within the enclosure. If additional drives (up to two more are supported) are desired, they must reside separately outside the box. The drives, by the way, can now be *doublesided* if desired, and will therefore provide up to *twice* the on-line access and storage (a function of the new disk controller peripheral card module) when used with double-sided floppy disks. TI has yet to announce the availability or price of double-sided drives, but I think we can safely assume they will be forthcoming before not too long . . .

This brings up the question of differences between the existing TI peripherals (in separate enclosures) and the first four new peripheral modules that have been announced. Like the new controller module (supporting double-sided, single-density 5¼" drives), the new RS-232 interface module also has an added capability: it now provides a *parallel* port for interfacing with printers. This will save users the expense of purchasing a separate RS-232 interface card for their particular printer (such as an Epson MX-80, MX-100), and opens up the possibility of using (with the proper cables) printers without the RS-232 add-on capability (such as the Epson MX-70 or other parallel-only printers that utilize what is widely referred to as a "Centronics-compatible"interface).

The two additional modules that were announced and demonstrated at the show were the P-Code Card (used for writing and running programs written in Pascal, TI Pilot, and other forthcoming p-System languages) and the Memory Expansion Card (required for TI LOGO, the new Editor/Assembler package, and the UCSD Pascal system). Neither of these card modules differ from the separate stand-alone units.

Will the new peripheral modules and TI-Box coexist with the present stand-alone peripherals? TI has indicated that the older style peripherals (including the *new*, separate P-code Peripheral) would be "available only until replaced by the peripheral cards." (The box and card modules are slated for release some time in the second quarter of 1982).

The significance of the TI-Box approach goes far beyond merely reducing the physical size of a fully configured system (formerly requiring a 6-foot desk!) to true desk-top proportions. For a start, the consumer will be able to save a significant amount of money as his system expands, since the card modules are (in some cases) quite a bit less money than their equivalent stand-alone units. In fact, the money *saved* when adding just a one-disk system and expansion memory will pay for the entire cost of the bare box (about $250 suggested retail). This cost savings is made possible by the elimination of separate power supplies and expensive enclosures that house the components and provide electromagnetic radiation shielding (to guard against TV interference, etc.). Incidentally, Texas Instruments has consistently chalked up the best record in meeting and exceeding the FCC's

**"...the TI–Box configuration ...leaves open three additional slots beyond the first four modules already announced, and opens up a whole range of exciting possibilities for new products ..."**

(Federal Communications Commission) radiation requirements, and has evidently designed this new system with the idea of upholding their reputation. That's why, in addition to the shielded modules, there's a thin, tightly sealed box-like connector that plugs into the right side of the computer console (or speech synthesizer, if attached) and connects it to the TI-Box (via a single shielded cable).

The other interesting aspect to the TI-Box configuration is its capacity to hold *seven* peripheral modules. This leaves open *three* additional slots beyond the first four modules already announced, and opens up a whole range of exciting possibilities for new products from both TI and various third-party suppliers. Speculation, anyone?

## The Elephant That Almost Never Forgets

Texas Instruments also used the occasion of CES to introduce another of its well-kept secrets—the *Mini Memory* Command Module. Although officially unveiled here, the "secret" of this revolutionary product unfortunately just didn't want to disseminate itself. Without the fanfare, attractive displays, and color sales literature that accompanied the TI-Box, I don't think there could have been more than a handful of visitors that really understood the significance of this product—one that TI must have cooked up in record time. After all, it simply looked like any other plug-in Command Module, carried somewhat of an ambiguous name, and didn't draw huge crowds to compete against hordes of attacking aliens (as did *TI Invaders*). But nevertheless, the significance of this "sleeper" will, I predict, be far ranging.

Although the additional user memory the module provides (4K bytes RAM) is considered "mini," the versatility that the TI-99/4 or TI-99/4A console gains by having it available is definitely MAXI! In addition to the memory—which is, by the way, no ordinary memory—the *Mini Memory* package also contains some very important tools for program development (housed in a 6K GROM and a 4K ROM), and allows TMS9900 Assembly Language programs to both be written and run on a "bare-bones" console—*without* any 32K-byte expansion memory or disk system. (Yes, programs *can* now actually be "assembled" in this minimum configuration, since purchasers of this package also receive a cassette-based line-by-line assembler!)

For the user who programs in TI BASIC, the module adds seven additional subprograms, including the ability to PEEK and POKE values from and to memory. The other subprograms are mainly concerned with linking to and being able to run Assembly Language programs and subroutines from TI BASIC. Additionally, if the 32K-byte memory expansion is attached, it can now be used by TI BASIC programs. (Before the only BASIC that could access the expansion memory was *Extended* BASIC.) The bottom line is

HOME COMPUTERS
NOW!...OVER 1000 SOFTWARE PACKAGES AVAILABLE
NOW!...OVER 1000 SOFTWARE PACKAGES AVAILABLE

a rejuvenated and now extremely flexible BASIC language—one that can revert to Assembly Language subroutines to perform functions which would be slow or inefficient to implement in BASIC alone.

Well, you've now seen what's in it for both Assembly Language and BASIC programmers, but what about all the *non*-programmers? What's so exciting about this new product? For the answer to that, you should first realize that many types (but definitely not *all*) of quick-responding and realistic applications and entertainment programs just *can't* be efficiently implemented in the BASIC language (TI's dialect or any other). With complex "high-level" English-like interpreted languages such as BASIC, it's usually a problem of speed, timing, or the inability to access certain built-in machine resources (due to the particular "architecture" of a computer). So naturally enough, there has been a noticeable shortage of this type of software (TI alone can do just so much . . . )

This is where Assembly Language and other "low-level" assembly-type languages do so well. This is why programs on TI Command Modules (written in assembly-level languages) are usually so impressive with their speed and graphics. At the risk of over-simplifying, the chips in these Command Modules provide a place for these low-level instructions to reside and operate from (without the need for the memory expansion peripheral). The *Mini Memory* Command Module does likewise—only with this little unit, you are able to sublet the *same* internal space to *many* program tenants: Whenever you tire of one, send'em away and bring in another. It's like having an *unlimited* number of Command Modules!

What this actually means is that users will soon be able to choose from a tremendous offering of high-quality software that has been programmed in Assembly Language. These programs will be sold by many third-party software developers and will be available on cassette tape. To run them, all that will be needed is the computer console and the *Mini Memory* module.

And now for the pièce de résistance, the crème de la crème, the cat's meow —all rolled up into one. Astute readers probably have some idea of what's coming. (After all, why bother with "cute" sectional sub-heads if they have no meaning . . . ) What I'm leading up to is the fact that this new user-RAM (random access memory) is indeed very special—RAM that "remembers." You can store either data or programs in the *Mini Memory* module and *not* have them erased (which happens to the normal RAM inside the computer) when the computer's power is turned off! The contents won't even be erased if you remove the module from the console! The module can later be snapped in and the program or data is *instantly* available.

TI accomplishes this memory trick with a special type of RAM (CMOS) that has a very low power consumption —allowing the chips to be powered by chargeable batteries (with a two-year shelf life) residing inside the Command Module cases, when not receiving power from the computer. Incidentally this memory is configured architecturally as "CPU memory," which means it is fast-access memory. Users will be able to take advantage of this fast storage and retrieval to store data which is used frequently in an application or to store Assembly Language programs which require a lot of computation. Also, if you have a small TI BASIC program that gets frequent use, you can store it in the *Mini Memory* module, rather than on a cassette tape or floppy disk, and benefit by virtually "instant" loading. [Watch for an in-depth article on using the *Mini Memory* module in the next issue.]

**"The module . . . contains important tools for program development . . . adds the ability to PEEK and POKE [and to call] Assembly Language subroutines . . . from TI BASIC."**

What's all this new capability and versatility cost? A mere $99.95 at suggested retail. Not bad, when you consider that some Japanese microcomputer manufacturers are selling bare-bones battery-packed "CMOS RAM packs" for their computers at over $150 per unit. And that's all you get—the 4K of memory. With TI's package you get the memory, the extra chips containing the utility routines, a cassette tape with an assembler and a graphic demonstration program, plus some very thorough documentation. Who says America can't compete?

## They're Finally Here!

If you've been following our articles in previous issues on the UCSD Pascal System and the Editor/Assembler package, you'll be glad to learn that these long-awaited products finally made their formal introductions at the show. The final implementations are ready, prices have been set, and TI will soon be shipping—the Editor/Assembler now, and the P-System hardware and software within a couple of months. The delivery of these products will open the floodgates to a massive release of additional software for TI personal computers. The number of high-quality applications and entertainment programs should increase geometrically over the next couple of years. (In anticipation of this phenomenal growth curve ahead, TI has already made plans for its future software directories to be compiled, updated, and produced in print-out format.)

For the serious programmer, TI's Editor/Assembler package has to be one of the greatest software values going . . . A suggested retail price of $99.95 buys the Command Module, two full disks of software (now including the source and object code for a very thorough interactive debugger), plus a huge 445-page owner's manual that provides extensive documentation of the TI-99/4A's software architecture. If all this wasn't enough, Texas Instruments has also thrown in (on one of the disks) as a programming example, the source and object code for their new arcade game, *Tombstone City*—a $39.95 suggested retail value (on Command Module) for the game itself!

TI's implementation of the UCSD Pascal System, Version IV.0, is another one of those accomplishments that get "lost" on the floor of a Consumer Electronics Show. For the average dealer, distributor, chain-store buyer, or member of the press, watching an operating system in operation isn't very exciting, to say the least. So it's only natural that this major step forward into the realm of software "portability" (being able to run virtually the same program on many different computers) and "user-friendliness" was largely overlooked.

What impresses me most about the Texas Instruments implementation of this popular operating system (Pascal is only *one* of the several languages available that run under the UCSD Pascal operating system) is the unique way that the hardware, firmware, and software have been "modularized"—both from a systems architecture and marketing standpoint.

The first place we start to see this uniqueness is in TI's attempt to bring the benefits of sophisticated Pascal-language software packages to non-programming end users who can't justify the purchase of floppy disk drives and expensive "systems software" that only programmers would need. With nothing more than a Memory Expansion Card ($299.95 suggested retail) and a P-code Card ($249.95) plugged into the TI-Box, users can run compiled Pascal software that they purchase on Command Modules or cassette tapes. This comprises a complete, efficient, and affordable p-System (at about *one third* the price of an minimal Apple II p-System) for end users who don't want to write their own software.

TI was able to implement this diskless p-System by having the pseudo-code interpreter and operating system burned into chips on the P-Code Card. This strategy not only kept the entry price-level low, but also produced a faster, easier to use p-System that typically handles the needs of programmers with *one less* disk drive than that of other systems. That's because the others depend on frequent disk swapping and overlaying to get around the annoyance of having to bring their additional disk-based software into and out of the limited RAM space.

Later, if the user needs disk storage, a Disk Controller Card ($249.95 suggested retail) and Expansion Box Disk Drive ($399.95) can be plugged into the TI-Box. If and when the ability to write programs becomes important to a user, TI has the "unbundled" software packages available separately: the UCSD-Pascal Compiler ($124.95 suggested retail), the UCSD p-System Assembler/Linker ($99.95) for writing TMS9900 Assembly Language programs, and the UCSD p-System Editor/Filer/Utilities ($74.95). Additionally, the soon-to-be forthcoming TI Pilot language has been announced ($79.95); this software is also designed to run under the TI p-System implementation.

## The Best is Yet to Come

This is the basis for my earlier statement about TI's unique dual "modularity"—how *both* consumers and software developers (with entirely different intended uses for the system) have to purchase *only* the hardware and software they actually need, and can expand their systems as desired at a later date. No other computer manufacturer presently provides this type of flexibility.

But what of the new Version IV.0 system itself? What does it offer programming students and advanced software authors? In a nutshell—the most flexible programming environment that is presently available for any microcomputer (including the highly-touted CP/M operating system). It allows software portability between more than 20 micros and minicomputers! In addition to being more portable than previous UCSD versions (and certainly *much* more portable than the numerous non-standard CP/M implementations), IV.0 offers new features and characteristics in the area of memory management and scheduling services, as well as important new options for the application program developer (such niceties as more code segments with fancier segment memory management, re-direction of standard inputs and outputs, command files, and program chaining).

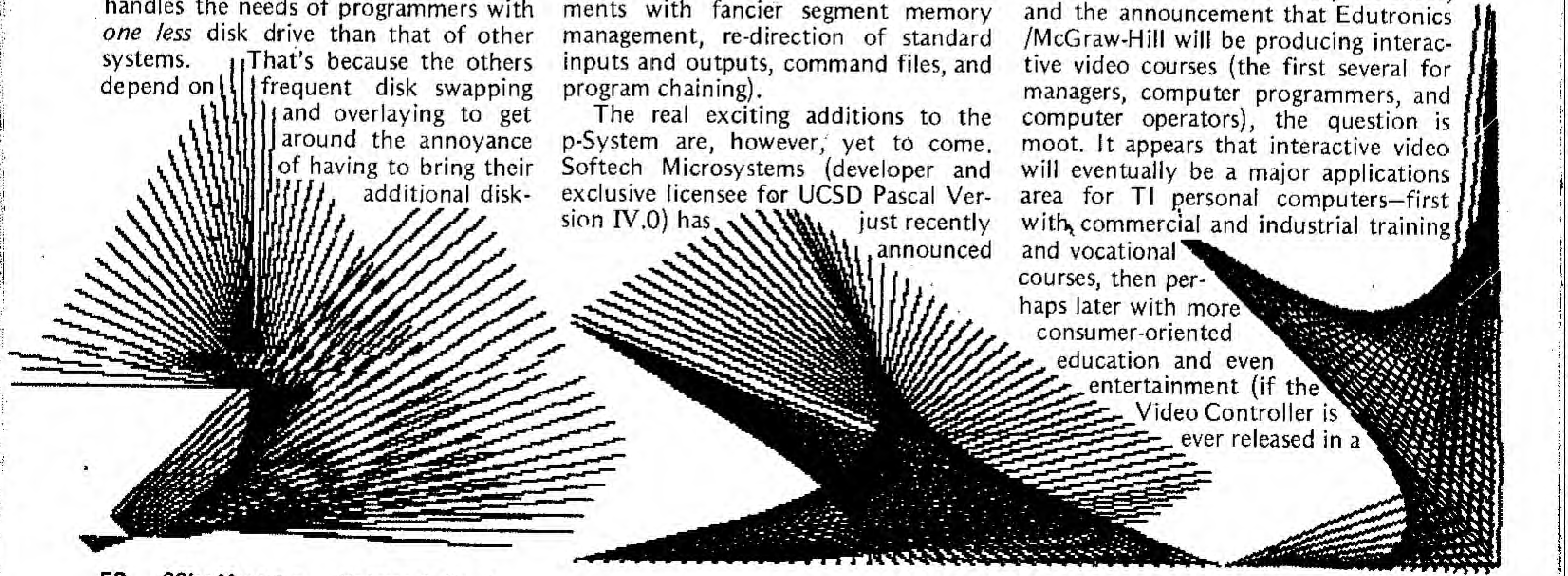
The real exciting additions to the p-System are, however, yet to come. Softech Microsystems (developer and exclusive licensee for UCSD Pascal Version IV.0) has just recently announced a series of add-on products for 1982—products that one hopes will find their way into the TI implementation. These include a print spooler, a computer-aided instruction package for UCSD Pascal, a 9900 native code generator, a CP/M data and text file access facility, and turtle graphics (for portable graphics software!).

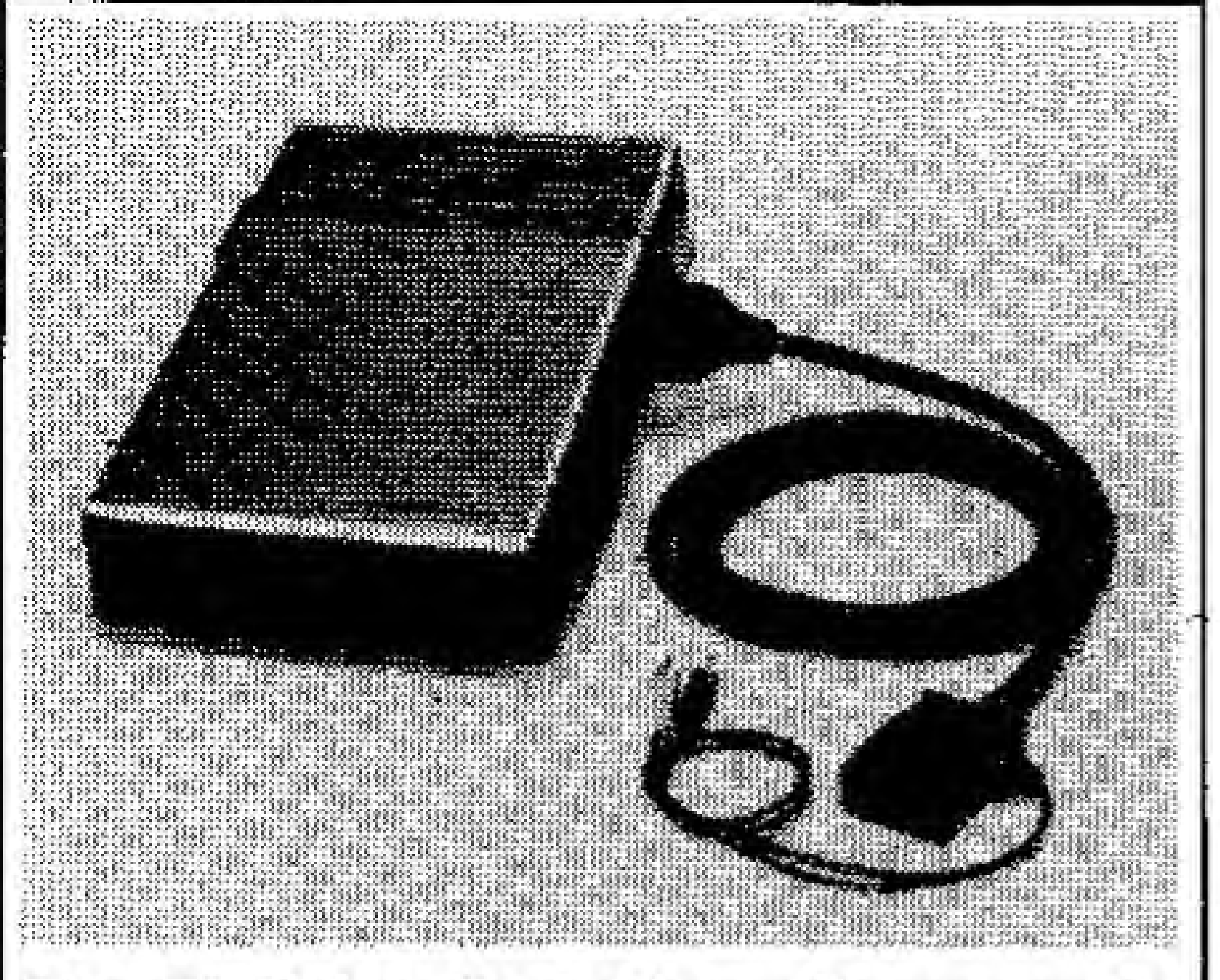
## A Bit of Graphic Magic

Visitors to the TI exhibit also had the opportunity to witness some impressive screen graphics. An Assembly Language program (called "Lines") loaded into the *Mini Memory* Command Module (and that initially comes on the same cassette tape as the line-by-line assembler) demonstrated the colorful, high-resolution "bit-map" graphic mode of the TI-99/4A. [An article explaining the TMS9918A Video Display Processor chip that makes both this and sprites possible will be in the next issue.] Then, with a few keystrokes, a "screen-dump" routine transferred the ever-changing line patterns to paper via the MPI 88G dot matrix printer (from Micro Peripherals, Inc.) that was interfaced to the computer. This printer was in its "dot addressable" graphics mode when it produced the pictures shown here. [For an explanation of how this mode of printing works, see *From Dots to Plots* in this issue.]

## The Video Controller Returns

Readers of my summer CES report (July/August p. 59) are already acquainted with the TI Video Controller (suggested retail, $699.95) and the Course Designer authoring software package ($199.95). A cable ($99.95) for interfacing with either a Panasonic or Sony VCR, or a Pioneer Video Disk Player is now also available. Last summer it was unclear whether the Consumer Products Group (CPG) would be marketing these products through regular distributor/dealer channels. Now, with these items listed in the CPG price sheet, and the announcement that Edutronics /McGraw-Hill will be producing interactive video courses (the first several for managers, computer programmers, and computer operators), the question is moot. It appears that interactive video will eventually be a major applications area for TI personal computers—first with commercial and industrial training and vocational courses, then perhaps later with more consumer-oriented education and even entertainment (if the Video Controller is ever released in a




**The TI Video Controller with connecting cable allows computer-controlled interactive video with VCRs and Video Disk Players.**

significantly less expensive card-module form that fits inside the TI-Box).

## Speech and Software Sensations

With each succeeding show, I naturally expect to see an increase in the amount of available software. This show, however, was the first time that TI publically displayed the current tally. And no wonder—the magic threshold of over 1,000 packages was finally reached!

Language software was in abundance, with five programming languages (TI BASIC, Extended BASIC, Assembly Language, Pascal, and TI LOGO) available for hands-on demonstrations. The TI LOGO language attracted quite a few interested visitors. This implementation of LOGO (the *first* and *only* one that has built-in sprite graphics) is a "natural" for product demonstration purposes since even a beginner can soon produce elaborate, animated color displays.

At last summer's CES, the innovative addition and subtraction software produced for TI by Scott, Foresman and Company caught my attention. This time around it was their new multiplication Command Module that impressed me. They seem to get better as they go along—which, I suppose, is how it should be. All you had to do was watch the faces of visitors (carrying shopping bags and literature from other microcomputer exhibits) while they tried some of this instant-loading CAI (computer-assisted instruction) math software, and you could obviously see that the TI/Scott-Foresman connection was working magic. I couldn't find any CAI software on the floor of the entire Convention Center that was in the same league with this math series. [See *99'er Magazine*; Sept./Oct., p. 64-67 for background material on this series.]

Finally, there were the games. If these weren't the hands-down winners in amassing visitor interaction time, I'd be really surprised—especially since everywhere you turned on the Convention Center floor, new video games visually and audibly "begged" for your participation. It was an arcader's paradise—no quarters necessary. TI had its new arcade games up and running: *TI Invaders*, *Carwars*, *Tombstone City*, *Blasto*, *The Attack*, and a prototype of *Munchman*. All consoles were equipped with the new TI Remote Controllers (joysticks). But due to the fact that there were so many video games on dedicated (non-computer) game machines, TI's offering didn't noticeably catch the fancy of the crowd—with one exception . . . *TI Invaders*. Virtually everyone who played it had to admit that it was better than any version of *Space Invaders* they had seen—even better than the original arcade version! I couldn't help noticing (especially among the Japanese who spawned the original game) the "instant addiction" to the TI product. I think that this game, more than anything, demonstrated to the crowds the fantastic potential of gaming on the TI personal computer.

## Cosby and Company on the Offense

Passersby to the exhibit were greeted by the familiar face and voice of TI spokesman, Bill Cosby—not in the flesh, but on a couple of dozen video monitors. The sales pitch was aimed at the present and potential sellers of the TI product line, describing all the media work Cosby would be doing during 1982. From the vast expenditure that will obviously be necessary to carry out this ambitious marketing plan, there's no doubt that Texas Instruments has finally started to launch its familiar price/performance/awareness blitzkrieg—we've already seen it in the past with digital watches and calculators—that will be orchestrated to ultimately leave competitors strewn by the wayside . . . But this time, unlike the watch and low-end calculator business that didn't in themselves lead to the development of other technological products , the more highly -competitive personal computer business fuels its own technological obsolescence and spawns innovation. And in the fast-paced decade of the 1980s, innovation will definitely be necessary for survival. It's here that TI has a leg up on its competition, for when it comes to technological innovation, very few can hold a candle to TI's awesome track record.

Perhaps that's why all the TI sales people on the floor were unusually active and aggressive. As the show wore on, and thousands upon thousands of visitor's flocked to their display area, I observed quite a few obviously weary TIers who seemed to be going on adrenalin alone . . . This was a far cry from the typical fatigue-induced "loafing" one observes at most exhibitors' displays during this type of long, grinding show. As orders were being taken, multitudes of eager new dealers signed up, and future commitments made, it was obvious that Texas Instruments enjoyed its new chance to bask in the limelight . . . and equally obvious that it would work extra hard to stay there. ■

## Want to Get Published?

**99'er Magazine** is looking for articles in all areas of interest that concern the Texas Instruments personal computers. Here are the kinds of articles that we want you to write for us:

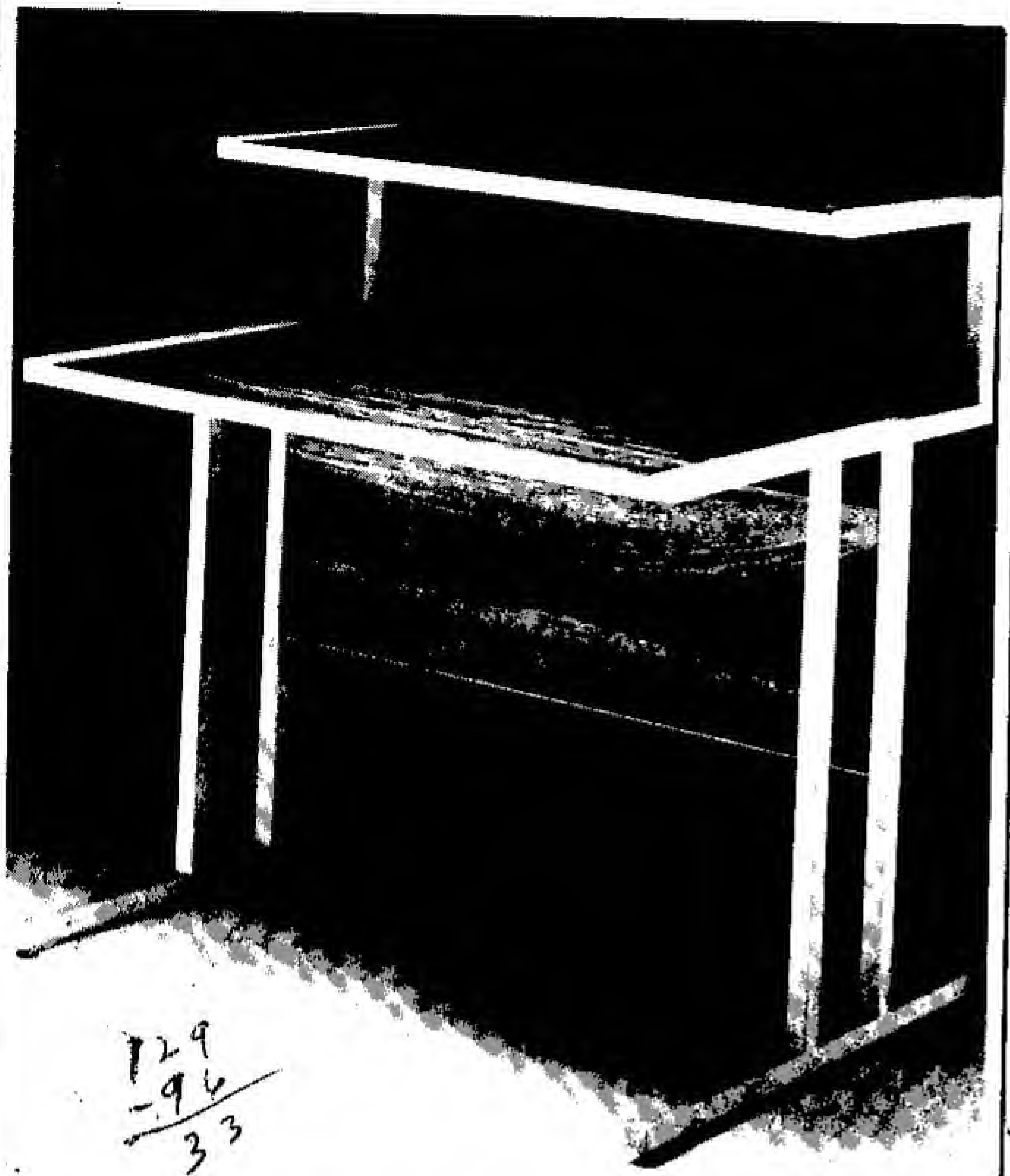
- Are you a businessman, professional, hobbyist, scientist, or engineer with an interesting microcomputer application? Tell us how it works, what problems you've had to overcome, and what recommendations you have for others. We're especially interested in sharing user-written software with our readers.
- Have you recently purchased a piece of hardware or software that hasn't quite come up to your expectations, or has, on the other hand, impressed you with its performance? We're looking for comprehensive product and book reviews from different perspectives.
- Are you an educator or parent with something to contribute to computer-assisted instruction (CAI)? We're always looking for new ideas and fresh approaches to educational problems.
- Have you created any unusual computer games or simulations? Let our readers experience your excitement and pleasure.
- Perhaps you've modified your microcomputer or have interfaced it with some unique or useful hardware. Send us your how-to-do-it story.

These are just some ideas. Perhaps you have others. Don't worry if you're not a professional writer. Our editorial staff stands ready to help polish up your manuscripts. And we'll be more than happy to send you a copy of our author's guidelines.

Please send your double-spaced typed manuscripts, plus disks or cassettes (recorded on both sides) if the article includes program material, to:

99'er Magazine / Editorial Dept.
2715 Terrace View Drive,
Eugene, Oregon 97405

## New Home Computer Work-Station Desk



## NOW You Can Separate Your TI Home Computer From Your Peripherals

This is a complete work station with a laminated top and a 3 ft. interconnect cable, which eliminates the "Freight Train Look".

**Choose from these three convenient sizes:**

- ● Model 24 Desk – 24W x 24D, top shelve 24W x 12D – **$250**
- ● Model 32 Desk – 32W x 24D, top shelve 32W x 12D – **$265**
- ● Model 40 Desk – 40W x 24D, top shelve 40W x 12D – **$280**

### Each Desk System Includes:

- 3 ft. interconnect cable, (connects 99/4 to peripherals)
- 6 outlet plug strip with on/off switch and re-set button,
- Bundle of tie-wraps to arrange wires on back of desk,
- Choice of colors-Honey teak, Walnut, Grey or Tan; all with blacks legs (Unless specified – Honey Teak, black legs standard)

*Note:* All top shelves are 35" from floor, and the work surface height is 26½ inches from floor.

The desk assembles without tools in minutes – All prices include freight pre-paid by HCS – Texas residents add 5% to amount for Sales Tax – Please allow 2-4 weeks for delivery – All orders will be collect unless payment is enclosed with order.

***HCS*** HARDIN'S COMPUTER SOLUTIONS INC.

P.O. BOX 634/RICHARDSON, TX 75080
838 N. GLENVILLE/RICHARDSON, TX 75081
214 699-8998 or 690-4942

How to Write BASIC . . . from p. 31

PROLITE
KI VAL

Table 1.
Condensed Format Code Table

| | | |
|---|---|---|
| 129 ELSE | 171 ??? | 213 LEN( |
| 130 : : | 172 ??? | 214 CHR$( |
| 131 ! | 173 ??? | 215 RND( |
| 132 IF V | 174 ??? | 216 SEG$( |
| 133 GO | 175 ??? | 217 POS( |
| 134 GOTO LN | 176 THEN LN | ( 218 VAL( |
| 135 GOSUB LN | 177 TO NE | 219 STR$( |
| 136 RETURN | 178 STEP NE | 220 ASC( |
| 137 DEF $V | 179 , | 221 PI |
| 138 DIM( | 180 ; | 222 REC |
| 139 END | 181 : $NV | 223 MAX( |
| 140 FOR NE | 182 ) $NE | 224 MIN( |
| 141 LET V | 5 183 ( | 225 RPT$( |
| 142 BREAK LN | 184 & $V | 226 ??? |
| 143 UNBREAK | 185 ??? | 22[illegible] |
| 144 TRACE | 186 OR NE | 22[illegible] [illegible] |
| 4 145 UNTRACE | 187 AND NE | 229 ??? |
| 146 INPUT #"": | 188 XOR NE | 230 ??? |
| 147 DATA N$ | 189 NOT =<> VE | 231 ??? |
| 148 RESTORE LN | 190 = NE $E | 232 NUMERIC |
| 149 RANDOMIZE | 191 < NE $E | 233 DIGIT |
| 150 NEXT | 192 > NE $E | 234 UALPHA |
| 151 READ V | 193 + NE | 235 SIZE |
| 152 STOP | 194 − NE | 236 ALL |
| 153 DELETE #N | 195 * NE | 7 237 USING |
| 154 REM | 196 / NE | 238 BEEP |
| 155 ON V | 197 ∧ NE | 239 ERASE |
| 156 PRINT "" or $ | 6 198 ??? | 240 AT( |
| 157 CALL PROGRAM | 199 Quoted string "" N | 241 BASE OPTION |
| 158 OPTION BASE | 200 Unquoted string N | 242 ??? |
| 159 OPEN #V | 201 Line number LN | 243 VARIABLE |
| 160 CLOSE #V | 202 EOF( | 244 RELATIVE |
| 161 SUB PROGRAM | 203 ABS( | 245 INTERNAL |
| 162 DISPLAY "" ; AT | 204 ATN( | 246 SEQUENTIAL |
| 163 IMAGE "$ | 205 COS( | 247 OUTPUT |
| 164 ACCEPT AT | 206 EXP( | 248 UPDATE |
| 5 165 ERROR | 207 INT( | 249 APPEND |
| 166 WARNING | 208 LOG( | 250 FIXED |
| 167 SUBEXIT | 209 SGN( | 251 PERMANENT |
| 168 SUBEND | 210 SIN( | 252 TAB( |
| 169 RUN | 211 SQR( | 253 # (files) NV |
| 170 LINPUT $V | 212 TAN( | 254 VALIDATE |

Table 1 together with their ASCII character tokens. In MERGE format, the BASIC elements listed are represented by a single ASCII character. For instance, CHR$(156) represents PRINT, CHR$(130) the statement separator, CHR$(213) the LEN function, etc. In order to prepare BASIC statements in MERGE format, however, one must also know to represent variable names, numeric and string constants, and line numbers occurring within statements.

The easiest of these to represent is the variable name; the normal ASCII representation for each character of the name is used. Consider the line:

```
10 PRINT XYZ
```

The MERGE format record used to represent this line would be:

```
CHR$(0)&CHR$(10)&CHR$(156)&"XYZ"&CHR$(0)
```

That is, seven bytes would be concatenated in a string and written in the appropriate disk file record. The first two bytes represent the line number; the next, the keyword PRINT; the next three, the variable name; and the last, the end-of-line mark. Assuming that the complete file corresponds to the requirements of MERGE format in other respects, when loaded into program memory with the MERGE command LISTing, the program will show it to contain the line intended.

Numeric constants and unquoted string constants are handled differently than variable names; each number of unquoted string *must* be preceeded by two identifying bytes. The first is CHR$(200), the character which signals the beginning of an unquoted string. Following CHR$(200), a byte must be included to indicate the number of subsequent characters in the string or number. This byte is simply the character with the code equal to the length of the string–i.e., if the string were five characters long, CHR$(5) must be included;

if 12 characters, CHR$(12). For example, consider the statement:

```
     10 PRINT X+345
```

The statement would be represented in MERGE format with 11 bytes as follows:

```
 CHR$(0)&CHR$(10)&CHR$(156)&"X"&CHR$(193)
    &CHR$(200)&CHR$(3)&"345"&CHR$(0)
```

Here, CHR$(200)&CHR$(3)&"345" first indicates that an unquoted string is to be encountered, then indicates how long that string is, and finally gives the string.

Quoted strings are handled in much the same way, except that CHR$(199) is used instead of CHR$(200).

```
  10 RUN "DSK1.FILENAME"
```

would be represented as:

```
 CHR$(0)&CHR$(10)&CHR$(169)&CHR$(199)&
    CHR$(13)&"DSK1.FILENAME"&CHR$(0)
```

Notice that quote marks are *not* explicitly included in the string representation. They are automatically provided for by the use of CHR$(199).

Finally, line numbers included in program statements such as GOTO and GOSUB must consist of two bytes coded in the same way as the line number bytes which begin each record. Moreover, these two bytes must be preceeded by CHR$(201) to indicate that they are to be interpreted as a line number. The statement:

```
     10 GOTO 200
```

would be represented as follows:

```
 CHR$(0)&CHR$(10)&CHR$(134)&CHR$(201)
    &CHR$(0)&CHR$(200)&CHR$(0)
```

## Program Generation

Although MERGE format programs can be generated with the above technique, its use--to say the least--would be cumbersome. The following method simplifies the process considerably.

For the moment, let's put aside the question of generating the portion of the character string associated with the BASIC statement. Assume that this string is generated and assigned to the string variable LINE$. Each time a LINE$ string is constructed, two line number bytes must be added to the beginning, an end-of-line byte to the end, and the whole thing must then be written as a record in the MERGE format file. The easiest way to handle the operations which follow the construction of LINE$ is to use a subroutine. Given a starting line number, LN, the following subroutine constructs the two-byte ASCII line number representation and writes the file record. It then increments the line number by 10:

```
9000  PRINT  #1 :  CHR$(INT(LN/256))&CHR$(LN–256*
      INT(LN/256) )&LINE$&CHR$(0)  : :  LN=LN+10  : :
      RETURN
```

After the BASIC statement portion of the record is assigned to LINE$, a simple GOSUB 9000 takes care of all the rest.

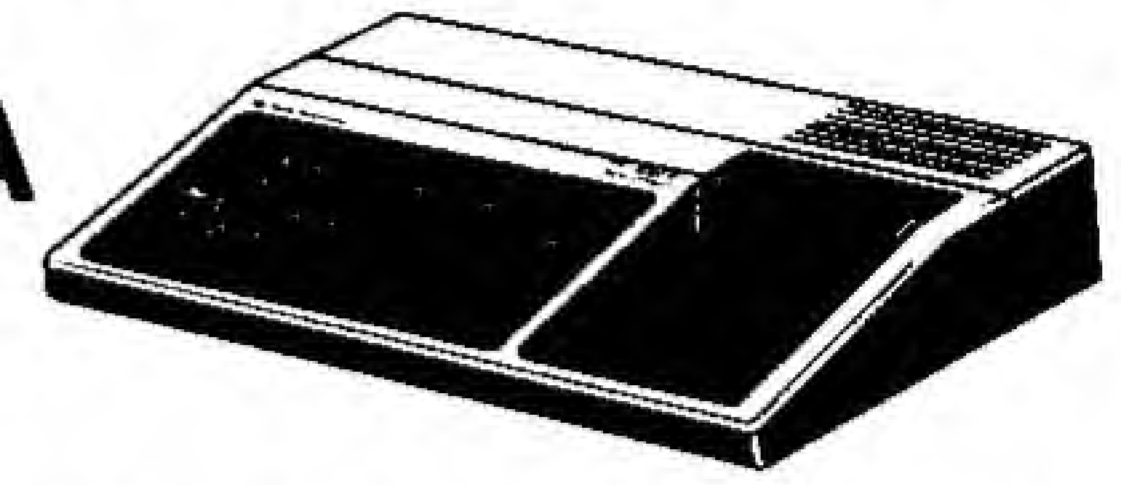
The construction of LINE$ strings can be simplified by assigning ASCII character codes to string variables with easy to remember names. For instance:

```
100  REM$=CHR$(154)::FOR$=CHR$(140): :NEXT$=
     CHR$(150)::IF$=CHR$(132)::THEN$=CHR$(176)
     ::TO$=CHR$(177)
```

Some string functions are followed by a "$" and are reserved words. But in TI BASIC, they can be imbedded in a variable name so that one could use variable names like @SEG$, @STR$, etc. for storage of the appropriate ASCII characters. Punctuation, arithmetic operators, and characters 199-201 also must be assigned "creative" string variable names; Q$ for quoted string, UQ$ for unquoted string, CM$ for comma, etc. or whatever will be easiest for you to remember.

**SAVE MONEY**
**ON**
**TEXAS INSTRUMENTS**
**PERSONAL COMPUTERS**

**TI-99/4A**
**$379**



| | |
|---|---|
| DISK DRIVE CONTROLLER | $229 |
| DISK MEMORY DRIVE | $369 |
| SOLID STATE PRINTER | $239 |
| 32K MEMORY EXPANSION | $239 |
| R.F. MODULATOR | $43 |
| TELEPHONE MODEM | $169 |
| RS 232 ACC. INTERFACE | $169 |
| CONTROLLERS (PAIR) | $31 |
| USER REFERENCE MANUAL | $9 |

**SOFTWARE**

| | |
|---|---|
| PHM 3007 Household Budget Management | 35.00 |
| PHM 3016 Tax/Investment Record Keeping | 60.00 |
| PHD 5003 Personal Financial Aids | 18.00 |
| PHD 5022 Business Aids Library-Finance Management | 34.00 |
| PHD 5024 Inventory Management | 60.00 |
| PHM 3005 Video Graphs | 18.00 |
| PHM 3020 Music Maker | 35.00 |
| PHM 3021 Weight Control and Nutrition | 53.00 |
| PHM 3023 Hunt The Wumpus | 22.00 |
| PHM 3030 A-Maze-Ing | 22.00 |
| PHM 3011 Speech Editor | 39.00 |
| PHM 3014 Statistics | 39.00 |
| PHD 5004 Programming Aids I | 13.00 |
| PHD 5005 Programming Aids II | 22.00 |
| PHD 5008 Electrical Engineering Library | 26.00 |
| PHD 5012 Programming Aids III | 18.00 |
| PHD 5013 Graphing Package | 18.00 |
| PHD 5016 Structural Engineering Library | 26.00 |
| PHT 6004 Programming Aids I | 9.00 |
| PHT 6008 Electrical Engineering Library | 23.00 |
| PHT 6013 Graphing Package | 13.00 |
| PHT 6016 Structural Engineering Library | 23.00 |

**CALL TOLL FREE**

| IN THE WEST | IN THE EAST |
|---|---|
| **(800) 648-3351** | **(800) 233-8950** |
| IN NV CALL | IN PA. CALL |
| (702) 588-5654 | (717) 327-9575 |

**COMPUTER MAIL ORDER**

**501 E. THIRD ST., WILLIAMSPORT, PA 17701**

**OVER 40 YEARS EXPERIENCE IN SOPHISITICATED ELECTRONICS**

Phone orders invited (800 number is for order desk only). Or send check or money order and receive free shipping. Pennsylvania residents add 6% sales tax. Add 3% for Visa or M.C. Equipment is subject to price change and availability without notice. Please call between 11 AM & 6 PM.

OTHER PRODUCTS NOW OR SOON AVAILABLE FROM DENALI DATA DESIGN

PROGRAMMING AIDS
- ★ SCREEN GRAPHICS LAYOUT SHEETS (INCLUDING SPRITE DEFINITION GRIDS)
- ★ MONITOR SCREEN OVERPLAY (GRID ON CLEAR FILM TO FIND WHERE THOSE CHARACTERS ARE REALLY AT.)
- ★ CODING SHEETS (BASIC & X-BASIC)

HARDWARE
- ★ ARTI-STIC™(JOYSTICK ADAPTER) ALLOWS THE USE OF FASTER, SMOOTHER-ACTING, LOW COST, ATARI™ JOYSTICKS ON TI 99/4 & 4A JUST PLUG IN AND PLAY
- ★ JOYPAD™ PLUGS INTO JOYSTICK CONNECTOR ON 99/4 & 99/4A AND ALLOWS DIRECTION & NUMERIC ENTRY IN ONE HAND HELD UNIT. (REQUIRES SUPPORT SOFTWARE — SUPPLIED IN BASIC AND X-BASIC ON TAPE OR DISK.)
- ★ 9 PIN & 25 PIN EXTENSION CABLES (SAME PLUGS AS ON JOYSTICK, CASSETTE & RS-232 PORTS.)
- ★ STACKER BUS™ (ALLOWS VERTICAL STACKING OF UP TO 4 PERIPHERAL DEVICES.)
- ★ BACKER BUS™ (ALLOWS PLACEMENT OF PERIPHERALS BEHIND CONSOLE.)
- ★ SPLIT-T BOX™(DOUBLE T-SWITCH FOR RS232. ALLOWS FOUR DEVICES TO BE PLUGGED INTO RS232 WITH HARDWARE SWITCHES TO ELIMINATE THE NEED FOR CABLE-SWITCHING.) (SOFTWARE SWITCHABLE AVAILABLE LATER THIS YEAR.)

SOFTWARE
- ★ CHECK BOOK RECONCILIATION (REQUIRES JOYPAD)
- ★ SCREEN TO EPSON MX80™ BLOCK GRAPHICS DUMP (REQUIRES JOYPAD)

Dealer Inquiries Invited

DENALI DATA DESIGN
ARTI-STIC™
JOYSTICK ADAPTER
Allows the use of faster, smoother-acting, low cost, Atari™ Joystick on TI 99/4 & 4A
JUST PLUG IN AND PLAY
*$20 postpaid*
Adapter + 2 Atari Joysticks
$35
Please add $2.00 for shipping

SEE YOUR DEALER OR CALL

Toll Free 1-800-654-8499
In OK dial 1-405-524-7764

1413 N. McKINLEY
OKC, OK. 73106

The next level of simplification involves use of user-defined functions to include more than one byte whenever possible. For example, it is clear that CALL will *always* be followed by an unquoted string; CALL COLOR, CALL SPRITE, CALL SOUND, etc. For that matter, the unquoted string token will always be followed by a byte indicating string length. Construction of strings which include the CALL keyword can therefore be simplified by defining functions appropriately:

110 DEF UQ$(X)=CHR$(200)&CHR$(X)::DEF [illegible]$(X)=CHR$(157)&UQ$(X)

A statement like CALL SCREEN(2) can then be written:

120 LINE$=CALL$( 6 )&"SCREEN"&LP$&UQ$(1)&"2"&RP$:: GOSUB 9000
(if CHR$(183), the left parenthesis, had previously been assigned to LP$ and 182, the right parenthesis, to RP$)

By making the function definitions a little more complex, the statement can be even further simplified:

110 DEF UQ$(X$)=CHR$(200)&CHR$(LEN(X$))&X$
120 DEF CALL$(X$)=CHR$(157)&UQ$(X$)

makes it possible to write CALL SCREEN(2) like this:

130 LINE$=CALL$("SCREEN")&LP$&UQ$("2")&RP$::GOSUB 9000

It's beginning to look a lot like BASIC.

Built-in functions can similarly be defined to facilitate construction of MERGE format strings. For instance,

140 DEF INT$(X$)=CHR$(207)&LP$&X$&RP$

allows one to write X=INT(Y/256) as:

150 LINE$="X"&EQ$&INT$("Y"&DIV$&UQ$("256"))::GOSUB 9000
(if CHR$(190) had been previously assigned to EQ$ and CHR$(196) TO DIV$)

Similarly, line numbers occurring within statements, such as GOTO or GOSUB can be simplified with the following function:

160 DEF LN$(X)=CHR$(201)&CHR$(INT(LN/256))&CHR$(LN−256*INT(LN/256))

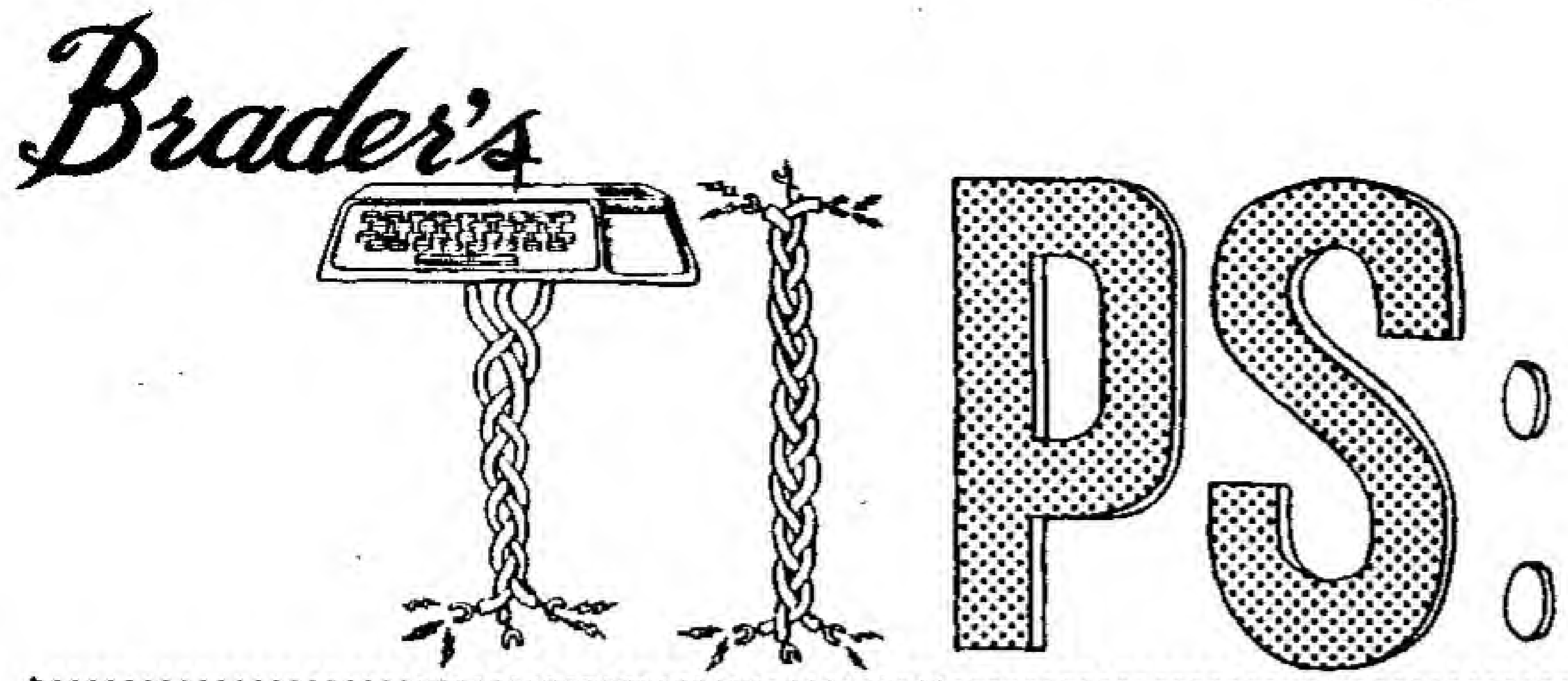
so that the statement GOTO 200 can be written simply as:

170 LINE$=GOTO$&LN$(200) :: GOSUB 9000
(if GOTO$ had previously been assigned CHR$(134)).

Using string variable names and user-defined string functions, you can create your own custom "language" for use in writing MERGE format records.

The following program may help to tie up the concepts presented; it is a trivial example of a music program generator. The program writes CALL SOUND statements in the MERGE format file "DSK1.BASIC" as the user presses a single key.

```
100 REM  **MUSIC PROGRAM GENERATOR**
110 REM  99'ER VERSION 1.4.1XB
120 REM  BY JOHN CLULOW
130 REM
140 REM ASSIGN STRING VARS
150 REM
160 CM$=CHR$(179):: LP$=CHR$(183):: RP$=CHR$(182)
170 REM
180 REM DEFINE STRING FUNCTS
190 REM
200 DEF UQ$(X$)=CHR$(200)&CHR$(LEN(X$))&X$ :
    : DEF CALL$(X$)=CHR$(157)&UQ$(X$)
210 REM
220 REM ASSIGN FIRST LINE NO
230 REM
240 LN=100
250 REM
260 REM OPEN MERGE FILE
270 REM
280 OPEN #1:"DSK1.BASIC",VARIABLE 163
290 REM
300 REM DISPLAY INSTRUCTIONS
310 REM
320 DISPLAY AT(7,1)ERASE ALL:"TO ENTER A NOTE PRESS ONE OFTHE FOLLOWING KEYS:"
330 DISPLAY AT(12,5):"A  B  C  D  E  F  G" :: DISPLAY AT(20,7):"PRESS P TO STOP"
340 REM
350 REM ACCEPT KEY INPUT
360 REM
370 CALL KEY(0,KEY,STATUS):: IF KEY=80 THEN 500 ELSE IF KEY<65 OR KEY>71 THEN 370
380 KEY=2*KEY-130 :: IF KEY>2 THEN KEY=KEY-1 :: IF KEY>7 THEN KEY=KEY-1
390 REM
400 REM COMPUTE FREQUENCY
410 REM AND PLAY NOTE
420 REM
430 FREQ=INT(440*(2^(1/12))^KEY+.5):: CALL SOUND(500,FREQ,0)
440 REM
450 REM FORM MERGE STATEM'T
460 REM
470 LINE$=CALL$("SOUND")&LP$&UQ$("500")&CM$&UQ$(STR$(FREQ))&CM$&UQ$("0")&RP$ :: GOSUB 490
480 GOTO 350
490 PRINT #1:CHR$(INT(LN/256))&CHR$(LN-256*INT(LN/256))&LINE$&CHR$(0):: LN=LN+10 :: RETURN
500 REM WRITE END-OF-FILE
510 REM
520 PRINT #1:CHR$(255)&CHR$(255)
530 CLOSE #1
540 REM
550 REM LOAD INSTRUCTIONS
560 REM
570 DISPLAY AT(12,1)ERASE ALL:"TO LOAD THE PROGRAM:" :: DISPLAY AT(15,2):"1) ENTER 'NEW'"
580 DISPLAY AT(17,2):"2) ENTER 'MERGE DSK1.BASIC'" :: DISPLAY AT(19,2):"3) ENTER 'RUN'" :: END
```



# TIMELY TROUBLESHOOTING & TINY TUTORIALS FOR YOUR TRIALS & TRIBULATIONS

*In this column, David Brader answers questions on any area of TI-99/4(A) computing. The most representative questions received will be answered and printed in this column. Do you have a question? Send it to:*

BRADER's TIps
99'er Magazine
P. O. Box 5537
Eugene, OR 97405

The usual trouble with the *first* edition of a question-and-answer column is the lack of questions! We're indeed fortunate not to be in that situation since many of you have faithfully responded to the bind-in questionnaire cards. We'll therefore draw our first batch of questions for this column from these.

## BASIC Pauses

Why does it often take so long for the cursor to come back after I change a line in a large program?

There are a couple of things going on during this pause: One thing happening is an adjustment of the length of the line you changed (if you shortened, lengthened, added, or deleted it). This may require repositioning all the lines that follow the one you changed. Another thing adding to the pause is the adjustment of any "pointers" in the program that may reference any lines that have been moved. These pointers are associated with BASIC statements like GOTO, GOSUB, NEXT, or IF-THEN.

Why does it sometimes take so long for a BASIC program to start after entering RUN?

This pause is what TI calls "prescan." During this time, the entire program is scanned and a list with all the program variable names and locations is generated in memory. This list is often called a "symbol table." Then, all variables are set to their initial values and the DATA statement pointer is set to the first DATA statement. [**Note:** The new release (version 2) of Extended BASIC provides you with the ability to control prescan by turning it on and off at various points in the coding, thus cutting down on lengthy pre-RUN waits—Ed.]

## BASIC "Garbage"

I noticed that sometimes a RUNing TI BASIC program seems to hesitate or stop for a few seconds. Then it starts up again like nothing happened. Is there something wrong with my computer?

Your computer is not sick. Believe it or not, this hesitation is a pause for "garbage collection"! Part of the TI-99/4(A) memory sometimes is used to hold data or text strings that the BASIC program is working on. Each time the program needs some of this memory for new data or new text, it uses some that it had not used before. Well after a while, all the unused or "clean" memory is gone. So the program is temporarily halted so that the "garbage" memory can be "cleaned" and "collected" into what is called the "free space pool." Then the program is allowed to continue.

## RS232 and External Printers

How can I connect a dot matrix or letter-quality printer to my TI-99/4A system?

This question comes up a lot. Currently, the only way to hook up a foreign device (other than floppy disk drives; see Sept./Oct. issue p. 48-51) to the TI-99/4(A) is through the TI RS232 Accessories Interface (model PHP1700). This will allow the addition of two devices (like a telephone modem and a printer). The device that is being connected must have an RS232 interface built into it. Finally, a cable must be purchased to connect the foreign device to the PHP1700. [**Note:** TI's new Peripheral Expansion Box recently unveiled at the Consumer Electronics Show (see related article this issue) will support a new plug-in RS232 Interface module with a *parallel* port in addition to the two RS232 serial ports. This will allow hooking up many popular printers directly, eliminating the need for purchasing a separate RS232 accessory card for printers that don't include one as standard—Ed.]

**TIP:** The Epson MX-80, MX-80FT, and MX-100 printers use a standard "RS232 male-to-male" type cable. For the Okidata Microline 82A, with its standard built-in RS232 interface, this is not true. Okidata puts the printer "busy" signal on pin 11 of the cable, and the TI RS232 Interface needs that signal on pin 20. So in this case, a standard RS232 male-to-male cable must be modified on one end by swapping the wires in pins 11 and 20. If this is not done, the printer will not print parts of some lines (due to data overrun).

## Word Processing

How can "word processing" be done with the TI 32-column display when most text is 80 or ever 132 columns wide?

Word processing with a 32-character line length is handled quite differently from a normal, commercial word processing system. There are currently two approaches to this problem:

The first approach is simply to allow the line to fold back to the left side of the screen each time you create a segment that is greater than 32 characters long. The trouble with this is the problem you have visualizing the actual format or layout that is being created. When you are trying to lay out something in a tabular format it often becomes a trial and error project. This approach, however, is easy to implement in BASIC and is fine for letter writing or normal text creation.

The second approach is more complex to implement (usually done in Assembly Language rather than BASIC) but does away with the folded lines on the screen. It uses a "window" into the text format that you are creating. This "window" is 24 lines high by 32 characters wide. If you cut a small square hole in a blank sheet of paper and place it over the text in this magazine, you will get the idea of this "window." By moving the sheet of paper up, down, left, and right, different portions of the text format are brought into view of the "window." You can similarly slide text on a TV display or monitor with the use of special keys on the TI-99/4(A) computer.

*Faster than a speeding alien laser,*
*More powerful than an 8-bit data train,*
*Able to leap tall sorts in a single bound . . .*
*Look — Up in the high RAM —*
*It's an editor . . . It's an assembler . . .*

# IT'S SUPER LANGUAGE!

## PART 2: Fundamentals of Assembly Language Programming on the TI-99/4A

**By Patricia Swift**
*Assembly Language Editor (The Human One)*

In the last issue we gave you a preliminary look at TI's new Editor/Assembler package for the TI-99/4 and 99/4A, and mentioned briefly the advantages of programming in Assembly Language. This article explores the benefits of Assembly Language more fully by comparing some programs written in Assembly Language and BASIC.

First, I would like to update two statements made in the previous article: The Editor/Assembler will now have a debugger (a modified version of an existing debugger, called TIBUG, that outputs to the monitor of the 99/4 and 99/4A). Our prototype does not yet include TIBUG, but it has been referenced in the latest documentation. Future articles will describe this debugger as soon as it is available. Also, I incorrectly called the macro VMBW a "Basic Support Utility." Actually, TI is calling it a "Utility Routine." The distinction is as follows: The Editor/Assembler package includes three groups of macros: (1) *Utility Routines*, for accessing machine resources, (2) *Extended Utilities*, for accessing routines built into the console ROM's and GROM's, and (3) *Basic Support Utilities*, for accessing the parameter list in CALL LINK statements from Extended BASIC.

### Some Assembly Language Explanations

Before examining some programs, it would be useful to mention some general characteristics of the TMS9900 processor, and then some specifics on the structure of the TI-99/4A.

All 9900 programs make use of 16 workspace registers, each containing 16 bits (one word). These registers reside in memory, and are pointed to by the hardware register called the Workspace Pointer. Most Assembly Language main programs define 16 contiguous words of memory for these workspace registers, and start out by setting the workspace pointer to point to that memory location. The fact that these registers reside *in memory* is one of the most powerful and unique features of the 9900-family processors. In an Assembly Language program, the current workspace registers are referenced by the hexadecimal numbers 0 through F. (In my program comments I will often call them R0 through R15.)

The structure of the memory of the 99/4A is fairly complex. The following explanations cover concepts necessary to understanding the programs in this article, but they only begin to scratch the surface of the memory structure.

CPU RAM (Random Access Memory) resides in the console and is directly addressable by Assembly Language programs. Workspace registers and other memory locations, as well as the programs themselves, reside in CPU RAM.

VDP (Video Display Processor) RAM, also located in the console, takes care of the video screen. Sprites, colors, character patterns, and the screen image itself all reside in VDP RAM. Unlike CPU RAM, however, VDP RAM is *not* directly addressable by Assembly Language programs. VDP RAM is accessed through specific assigned CPU RAM addresses. This is called "memory mapping." Locations 0 through>02FF (the symbol > means hexadecimal notation, and 02FF = 767 in decimal) in VDP RAM contain the screen image. This means that whatever characters reside in this section of VDP RAM are visible on the screen. To change the screen, the programmer would place the desired character code(s) into VDP RAM at the corresponding location(s). VDP RAM location 0 corresponds to the home position (upper left) on the screen; location 48 (or > 30) corresponds to the position called row 2 and column 17 in BASIC. Let's say you want to put an * on the screen at row 2, column 17. The ASCII code for * is 42, or >2A, and the desired VDP RAM location is >30. You might be tempted to use a MOVB (Move Byte) instruction to accomplish this, but remember that VDP RAM cannot be directly addressed from your Assembly Language program. To access VDP RAM, you'll need to use a Utility Routine. And in this case of moving a single tyte, the routine called VSBW (VDP Single Byte Write) is appropriate. VSBW is a macro which places the most significant (leftmost) byte of workspace register 1 at the VDP RAM address contained in register 0. Therefore, to place the * at row 2, column 17, you'd code:

```
REF     VSBW      UTILITY REFERENCE
 .
 .
LI      0,>30     R0 = VDP RAM ADDRESS
LI      1,>2A00   R1 CONTAINS  *  IN MSB
BLWP    @VSBW     MOVE TO VDP RAM
```

Most of the utilities use similar schemes of loading data into certain registers and calling the utility by name. I'll talk more about some specific ones later.

### The Game of Life

*Life* is a classic computer game. It is based on the idea of a population which goes through life cycles to form new generations; each position on the screen corresponds to a cell in the population. Cells which are alive are filled in (with asterisks in my example); dead cells are blank. The life cycle, or rules of the game, are applied to each generation to obtain the next generation, and then the new generation is displayed on the screen. The rules of the game determine birth, death, or survival of individual cells, and depend on the state of each cell's 8 neighbors (adjoining cells, considered horizontally, vertically, and orthogonally) as follows:

1. A live cell with 2 or 3 neighbors survives to the next generation.
2. A live cell with 0 or 1 neighbor dies of loneliness; a live cell with more than 3 neighbors dies of overcrowding.

## LIFE LISTINGS — Assembly, Listing 1.

```
       IDT  'LIFEA
       DEF  LIFEA
       REF  VMBW
WS     BSS  32
SCRN   BSS  768
GENSCR BSS  768             SCREEN IMAGE
OFSET  DATA -33,-32,-31,-1 OFFSETS OF NEIGHBORS
       DATA 1,31,32,33
FSTGEN DATA 7,335,366,368,397,401,429,433
H00    BYTE >00
H01    BYTE >01
H02    BYTE >02
BLNK   BYTE >20
STAR   BYTE >2A
AFTER  BYTE 0,1,1,0
       EVEN
H2000  DATA >2000
LIFEA  LWPI WS              START OF PROGRAM
* CLEAR SCREEN ARRAY.
       LI   1,766           LOOP COUNTER & INDEX
CLEAR  CLR  @SCRN(1)        CLEAR WORD
       DECT 1               POINT TO NEXT WORD
       JLT  INIT            DONE
       JMP  CLEAR           LOOP
* LOAD INITIAL GENERATION AND DISPLAY.
INIT   MOV  @FSTGEN,3       R3 = # OF CELLS
       A    3,3             DOUBLE IT FOR WORDS
INITLP MOV  @FSTGEN(3),4    R4 CONTAINS OFFSET
       MOVB @H01,@SCRN(4)   SCREEN POSITION = 1
       DECT 3               NEXT
       JNE  INITLP          MORE TO DO
       BL   @SHOWIT         SHOW INITIAL GEN
       LIMI 2               ENABLE VDP INTERRUPT FOR QUIT
* CALCULATE NEXT GENERATION.
CLCGEN LI   1,33            INDEX (ISUB)
       LI   3,22            OUTER LOOP CTR (ROW)
CLCLP  LI   4,30            INNER LOOP CTR (COL)
* COUNT NEIGHBORS.
CLCNBR LI   5,0             NEIGHBORS COUNTER (CNT)
       LI   6,0             LOOP CONTROL, INDEX TO OFSET
NBRS   MOV  1,7             COPY TO WORK ON
       A    @OFSET(6),7     R7->DISP OF NEIGHBOR
       CB   @SCRN(7),@H00   NBR=0?
       JEQ  NXTNBR          YES
       CB   @SCRN(7),@H02   NBR=2?
       JEQ  NXTNBR          YES
       INC  5               NEIGHBOR ON
NXTNBR INCT 6
       CI   6,16            DONE?
       JLT  NBRS            LOOK AT NEXT NEIGHBOR
       CB   @SCRN(1),@H01   IS CELL NOW ON?
       JEQ. CELLON          YES
       CI   5,3             3 NEIGHBORS?
       JEQ  CHANGE          YES-BIRTH
       JMP  NOCHG           NO
CELLON CI   5,2             2 NEIGHBORS?
       JEQ  NOCHG           YES-SURVIVE
       CI   5,3             3 NEIGHBORS?
       JEQ  NOCHG           YES-SURVIVE
CHANGE AB   @H02,@SCRN(1)   BIRTH OR DEATH
NOCHG  INC  1               NEXT CELL
       DEC  4               NEXT COL
       JNE  CLCNBR
       INCT 1               SKIP 2 EDGE CELLS
       DEC  3               NEXT ROW
       JNE  CLCLP
* RESET SCRN ELEMENTS TO 0 FOR DEAD, 1 FOR ALIVE.
       LI   5,33            INDEX TO SCRN (ISUB)
       LI   3,22            ROW CTR
LOOP   LI   4,30            COL CTR
LOOP1  MOVB @SCRN(5),6      R6=CELL VALUE IN MSB
       SRL  6,8             SHIFT TO LSB
       MOVB @AFTER(6),@SCRN(5) CHANGE CELL TO 0 OR 1
       INC  5               NEXT CELL
       DEC  4               NEXT COL
       JNE  LOOP1
       INCT 5               SKIP EDGE CELS
       DEC  3               NEXT ROW
       JNE  LOOP
       BL   @SHOWIT         SHOW NEW GENERATION
       JMP  CLCGEN          CALC NEXT GEN
* SUBROUTINE TO DISPLAY GENERATION ON SCREEN.
SHOWIT LI   5,767           R5 INDEXES BOTH SCRN
                            & GENSCR
BLDSCR CB   @H00,@SCRN(5)   IS BYTE 0 (DEAD)?
       JEQ  BLK             YES
       MOVB @STAR,@GENSCR(5)  NO-PUT * IN GENSCR
       JMP  NXTPOS
BLK    MOVB @BLNK,@GENSCR(5)  PUT BLANK IN GENSCR
NXTPOS DEC  5               POINT TO NEXT CELL
       JLT  OUTSCR          DISPLAY IF DONE
       JMP  BLDSCR          LOOP IF NOT DONE
OUTSCR CLR  0               VDP RAM ADDRESS (HOME)
       LI   1,GENSCR        GENSCR CONTAINS DISP DATA
       LI   2,768           768 BYTES TO WRITE
       BLWP @VMBW           WRITE SCREEN
       B    *11             RETURN
       END  LIFEA
```

## BASIC, Listing 2.

```
100 CALL CLEAR
110 DIM OFFSETS(8),AFTER(4)
120 FOR I=1 TO 8
130 READ OFFSETS(I)
140 NEXT I
150 DATA -33,-32,-31,-1
160 DATA 1,31,32,33
170 DIM SCRN(768)
180 REM  INITIALIZE
190 FOR I=1 TO 768
200 SCRN(I)=0
210 NEXT I
220 AFTER(0)=0
230 AFTER(1)=1
240 AFTER(2)=1
250 AFTER(3)=0
260 REM  INITIAL POPULATION
270 READ NUMSUB
280 FOR I=1 TO NUMSUB
290 READ ROW,COL
300 ISUB=(ROW-1)*32+COL
310 SCRN(ISUB)=1
320 CALL HCHAR(ROW,COL,42)
330 NEXT I
340 DATA 7
350 DATA 11,16,12,15,12,17,13,14,13,18,
    14,14,14,18
360 REM  CALCULATE NEXT GENERATION
370 ISUB=34
380 FOR ROW=2 TO 23
390 FOR COL=2 TO 31
400 CNT=0
410 FOR K=1 TO 8
420 M=SCRN(ISUB+OFFSETS(K))
430 IF M=0 THEN 460
440 IF M=2 THEN 460
450 CNT=CNT+1
460 NEXT K
470 IF SCRN(ISUB)=1 THEN 500
480 IF CNT=3 THEN 520
490 GOTO 530
500 IF CNT=2 THEN 530
510 IF CNT=3 THEN 530
520 SCRN(ISUB)=SCRN(ISUB)+2
530 ISUB=ISUB+1
540 NEXT COL
550 ISUB=ISUB+2
560 NEXT ROW
570 REM  SHOW NEW GENERATION
580 CALL CLEAR
590 ISUB=34
600 FOR ROW=2 TO 23
610 FOR COL=2 TO 31
620 SCRN(ISUB)=AFTER(SCRN(ISUB))
630 IF SCRN(ISUB)=0 THEN 650
640 CALL HCHAR(ROW,COL,42)
650 ISUB=ISUB+1
660 NEXT COL
670 ISUB=ISUB+2
680 NEXT ROW
690 GO TO 370
700 END
```

3. An empty cell with exactly 3 live neighbors is born (becomes alive) in the next generation.

The rules are applied to a generation as a whole, before the next generation is displayed. Depending on the initial population, you may see a colony which goes on changing forever, one which dies out or becomes static after a few generations, or one which oscillates among a few patterns.

There are a few restrictions on my implementation of *Life* which should be explained. First, I have defined the initial population in the programs, whereas other versions might allow the user to enter the initial population on the screen at the beginning of the game. In order to be sure the colony does not exceed the size of the 99/4A screen, which is 32 x 24, I have forced the border (row 1 and 24 and columns 1 and 32) to always remain blank. This means that when the colony becomes large it may lose its symmetry as one side of the colony hits the border.

The two programs which follow are in Assembly Language and in BASIC. Both follow the same strategy: display the initial colony, calculate the next generation by considering the neighbors of each cell in turn, clear the screen, display the new generation, and loop back to calculate the next generation. The Assembly Language version uses one byte to represent each cell; the BASIC version uses one entry in array SCRN for each cell. At the start of each generation, live cells contain the value 1 and dead cells contain 0. During the calculation of the next generation, a cell can have the values 0 through 3 as follows:

| | | |
|---|---|---|
| 0 | = | cell is dead and remains dead for next generation |
| 1 | = | live cell survives to next generation |
| 2 | = | dead cell will be born in next generation |
| 3 | = | live cell will die in next generation |

It is necessary to have these four possible values during the calculation so that the program can have information about the current state of each cell while calculating and storing the next state of each cell. Just before the new generation is displayed (or not displayed if dead), the values of the cells are reset to 0 or 1 by means of the array AFTER.

In examining both versions of *Life* which follow (Listings 1 and 2), you might wonder why anyone would use the more esoteric Assembly Language over the easier-to-understand BASIC. The answer is simple—*speed*. On the 99/4A, the BASIC program takes 2 minutes and 26 seconds between generations; the Assembly Language program takes *less than one second!* The BASIC version is no fun at all to watch, whereas the Assembly Language program provides fine entertainment. [The use of the Utility Routine VMBW (VDP Multiple Byte Write) in the Assembly Language version is partly responsible for this speed. It shows each new generation all at once. And fortunately, the monitor program is smart enough to capitalize on this by showing only the *changed* portions of the screen, rather than re-drawing the *whole* screen each time. If done fast enough, the human brain's "persistance of vision" allows us to see individual frames of moving images as *continuous* rather than *discrete* pictures—thus making realistic animation sequences truly possible—Ed.]

### Using Assembly Language to Move Sprites

The ability to create sprites which move automatically is one of the best features of the 99/4A. Sprites can be used in Extended BASIC and in Assembly Language programs.

VDP RAM has several areas dedicated to sprites. The Sprite Attribute Block, which gives the sprite locations, sprite numbers, and colors, starts at address>300. Each entry in the sprite Attribute Block occupies four bytes. A terminator byte with value>0D denotes the end of the Sprite Attribute Block. The Sprite Descriptor Block contains the sprite pat-

**KILL SURGES LIKE LIGHTNING!**

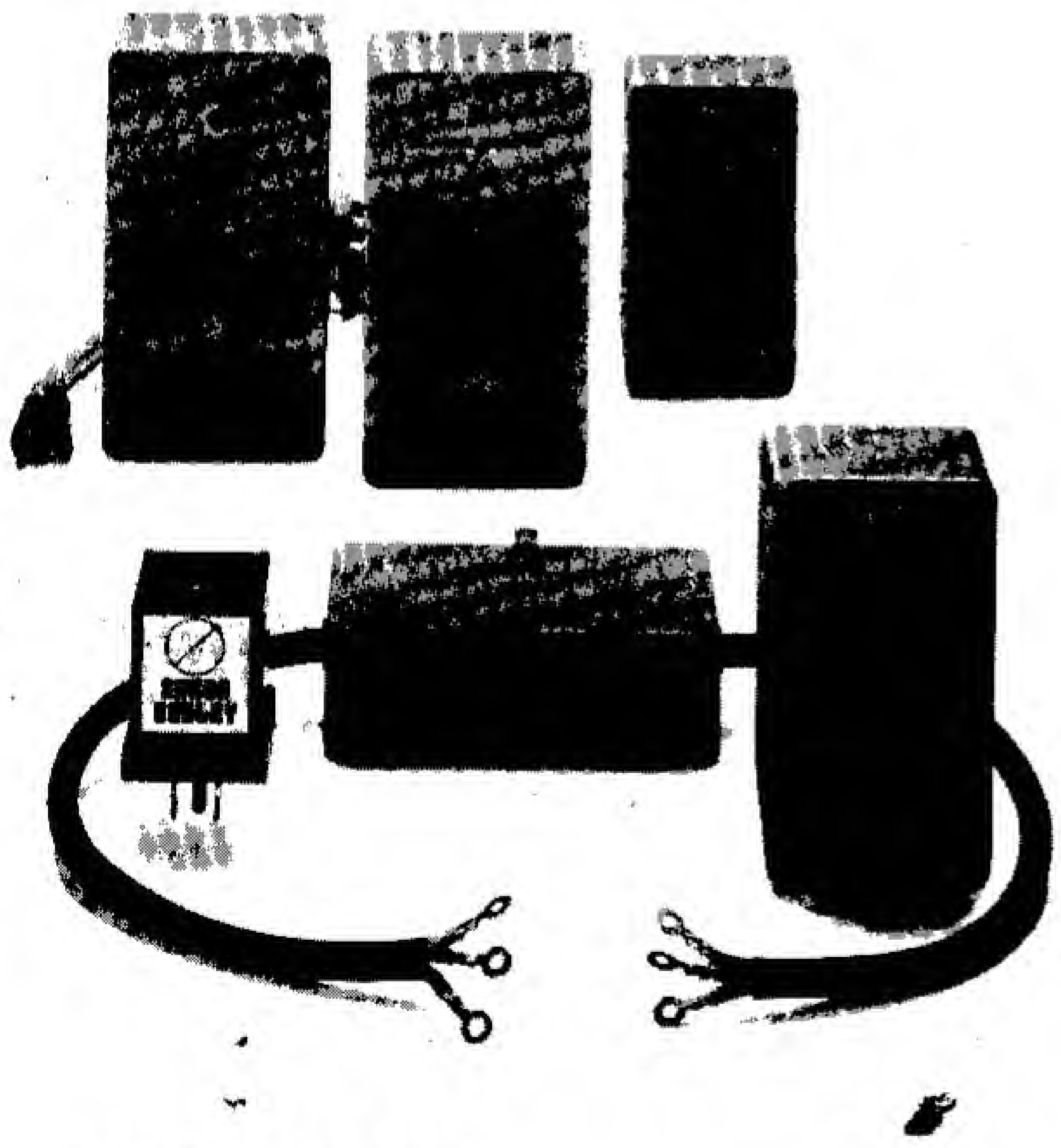
AC power line surges are destructive, can cost you money, and can't be prevented. But you can stop them from reaching your sensitive electronic equipment with a Surge Sentry.

Surge Sentry acts in picoseconds to dissipate up to a 1,000,000 W, 100μ second surge. Triggers at 10% above nominal peak voltage. Works in parallel with the power line. Is easy to install for immediate protection. No complicated wiring or special tools required.

Several different models to choose from, including an OEM version. Call or write today for a *free* brochure.

**SURGE SENTRY**

*It'll clean up your AC*



**RKS INDUSTRIES**

**208 Mount Hermon Road, Suite #2, Scotts Valley, CA 95066 • (408) 438-5760**

terns (shapes), with 8 bytes for each possible sprite. Although the Sprite Descriptor Block starts at VDP RAM address 0 by default, we have already seen that VDP RAM locations 0 through>02FF are used for the screen image table and locations >0300 through>03FF for the Sprite Attribute Block. In order to avoid writing over these areas, the Sprite Descriptor Block usually starts at location>0400 for practical purposes. The entries in the Sprite Descriptor Block are defined to correspond to sprite numbers starting at 0 and occupying 8 bytes each; therefore the entry at location>0400 is for sprite number>80. Thus in Assembly Language programs, the lowest sprite number is usually>80. The Sprite Motion Table, which gives the Y- and X-velocities of defined sprites, resides at VDP RAM location>0780. Each entry in the Motion Table occupies four bytes, the last two of which are for system use. The Sprite Motion Table is filled only if automatic motion is to be used. An Assembly Language program could move the sprites (non-automatically) by changing the Y- and X-locations of the sprites in the Sprite Attribute Block. But the system is able to move the sprites for you via interrupt processing routine: each time a VDP interrupt occurs (60 times per second), the interrupt processing routine moves any eligible sprites according to the Sprite Motion Table. In order to make use of this facility, the Assembly Language program must also load the number of moving sprites at CPU RAM address>837A and enable the VDP interrupts.

## Assembly VS Extended BASIC

You are probably thinking that this sounds like a lot of work to achieve moving sprites, especially compared to the simple CALL SPRITE statement of Extended BASIC. However, there are times when an Extended BASIC program is inadequate. Coincidence checking in Extended BASIC is not always accurate, especially if the sprites are moving fast. Extended BASIC is not as responsive to velocity changes as you might like.

The programs which follow (Listings 3 and 4) illustrate how Assembly Language can be used to overcome these deficiencies. The program simply moves a target from left to right on the screen while shooting an arrow from the top of the screen to the bottom. Both sprites wrap around the screen. Whenever the arrow hits the target, the sprites stop moving, the target changes to an X, and the program delays long enough to make the blow-up visible. Then the program starts over. The Extended BASIC program relies on CALL COINC to detect hits. You'll notice, however, that the program doesn't seem to detect all hits. The Assembler Language program can stop the action by disabling the VDP interrupt while it checks for coincidence by comparing the locations of the arrow and the target from the Sprite Attribute Block. Moreover, the Assembler Language program can check the point of the arrow against the target instead of checking the upper lefthand corners of the sprites.

Because of these differences, the Assembler Language program appears to detect more hits correctly. Of course, this stop-motion processing must slow down the motion, but it is not noticeable to me. (One indication of the speed of Assembler Language program execution is the large number of statements executed in LOOP2 while the hit shape briefly remains on the screen.)

Another shortcoming of the Extended BASIC version is that the hit shape appears quite a bit to the right of its actual position when the hit occurred. This is because the sprites have continued to move while two BASIC statements (lines 190 and 200) are interpreted and executed. The Assembly Language version has already stopped the motion by disabling the VDP interrupt program via LIMI 0; it doesn't start the motion again until after the hit sequence is complete. Thus, only the Assembly Language program actually shows the blow-up in the right place on the screen.

Continued on p. 76

## TI-PWRITER

*a complete WORD PROCESSOR for the 99/4 & 99/4A*

**ANY Printer** — T.P., RS232C (or screen only).
**FULL upper & lower case (including 99/4)** — even on screen print; lower case entry on 99/4A.
**Input typing SPEED** — over 100 words per minute.
**ANY Line Length** — 28 to 254 characters per print line.
**Complete text EDITING** — by cursor control; including insert & delete lines, partial text print, printer halt or abort without text loss, page FWD & BKWD, and automatic line centering.
**ANY Input/Output storage of text** — disc, cassette, cassette input/disc output, or vice versa.
**Holds 3000 characters (before storage)** — 50,000 characters per disc or 60 minute cassette.
**COMPLETE Software Control of Printer** (depending upon its capabilites) — for enhanced print, underlining, formatting, variable cpi.
**No Special Equipment** — monitor, console, Extended Basic module, storage device (preferably), printer (or a friend with one). Comes with a 20 page instruction booklet.
**CASSETTE**: $32.00/**DISKETTE**: $35.00

## ★ Extended BASIC ★ GAMES

**Extended Baseball** — full sprite action, dual joystick control (joysticks required).
**Extended Hangman** — 580 words, good graphics, speech optional.
One free program with each of the above:
**Gorfia Pestulitis** — fight the invading space monsters (joysticks required).
**Tic-Tac-Toe** — 4th level of play against computer is almost unbeatable (speech optional).
**CASSETTE**: $9.95 for 2/$16.95 for all 4
**DISKETTE**: $12.95 for 2/$19.95 for all 4

*• Extended BASIC Module Required For ALL Programs •*
If you are not *completely* satisfied you may return them within 15 days for a *full* refund of purchase price.
*Send check to:* **Extended Software Company**
11987 Cedarcreek Drive
Cincinnati, Ohio 45240
(513) 825-6645

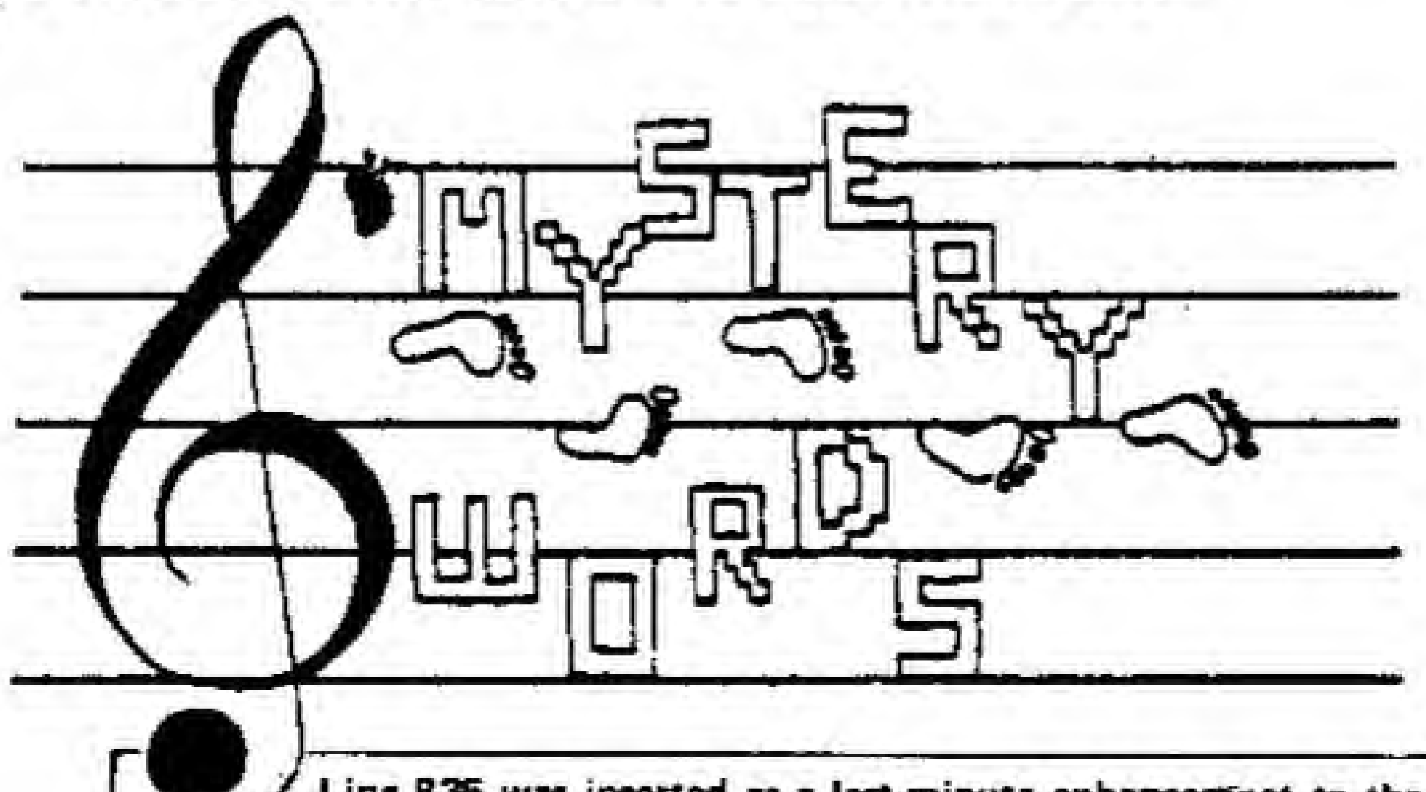
## Interplanetary . . . from p. 35

```
470 DATA "^^!!!^^______$_^^^!!!^^__"
480 DATA "^!!!!^^__$_$$$____^!!!^^_"
490 DATA "^!!!!!^_$$$____$$_^^^!!^_"
500 DATA "^!!>!!^^_$$$$_$$$$_^^!!^^"
510 DATA "!!!!!!^___$__$$___^^^!^^^"
520 DATA "!!!!!!!!^^_^^^_$__^^!!^___"
530 DATA "^^^!!!!!!^^^^__$_^^!!^__$$"
540 DATA "^^^!^^!^___^__$__^^!^_$$?"
550 DATA "_^^^^^^^__$^_^_$__^!^_$??"
560 DATA "$__^^^^^_$$^^^$$___^^__$?"
570 DATA "?$_^__^_$$$^^??$$__^^^__$"
580 DATA "?$^^_____$$__??$$__^_^^_"
590 DATA "???^^^___$$$$^^^^_^^^^!^_"
600 DATA "???_^^__$$$$$$$_^!^^^!!^_"
610 DATA "??$_^_$$$$???$$^!!^_^!!^^"
620 DATA "___^^__$$$???$_^!^_^^!!!!"
630 DATA "___^^^___$$?$_^^!^_^!!!!!"
640 DATA "___^^_^___$$__^!!!!!!>!!^"
650 DATA "__^__$___^_$_^!!^^^!!!!!^"
660 DATA "____^^^^__$_^^!!^^_^^^!^_"
670 DATA "______^^^__$_^^^^_$___^_$"
680 RETURN
690 DISPLAY AT(22,1):"ALT "
700 DISPLAY AT(23,1):"HVEL"
710 DISPLAY AT(24,1):"VVEL"
720 DISPLAY AT(22,15):"TIME"
730 DISPLAY AT(23,15):"FUEL"
740 DISPLAY AT(24,15):"PWR "
750 CALL VCHAR(6,30,63,4):: CALL VCHAR
    (10,30,42,4):: CALL VCHAR(14,30,95,4)
760 CALL VCHAR(18,30,94,4):: CALL HCHAR(22,28,33,3)
770 CALL SPRITE(#2,98,2,160,222,#1,96,2,D3,D4)
780 RETURN
790 CALL KEY(1,K1,S1):: CALL KEY(2,K2,S2):: IF S1=0
    AND S2=0 THEN 920
800 IF S1=0 THEN 860
810 IF K1=5 THEN D1=D1-1 :: E=E-50 :: GOTO 860
820 IF K1=0 THEN D1=D1+1 :: E=E-50 :: GOTO 860
830 IF K1=2 THEN D2=D2-1 :: E=E-50 :: GOTO 860
840 IF K1=3 THEN D2=D2+1 :: E=E-50 :: GOTO 860
850 IF K1=11 THEN F=TOFF
860 IF K2=3 THEN F=F-5000 :: GOTO 920
870 IF K2=12 THEN F=F-10000 :: GOTO 920
880 IF K2=2 THEN F=F-1000 :: GOTO 920
890 IF K2=5 THEN F=F+1000 :: GOTO 920
900 IF K2=6 THEN F=F+5000 :: GOTO 920
910 IF K2=11 THEN F=F+10000
920 IF E<=0 THEN E=0 :: F=0
930 IF F<0 THEN F=0
940 IF F=0 THEN CALL PATTERN(#3,32)ELSE CALL
    PATTERN(#3,104)
950 V2=F/(5+E)-G :: V=V+V2 :: DV=V :: IF V<0 AND
    H<=0 THEN DV=0
960 D=(V1+(V2/2)):: V1=V :: H=H+D :
    : E=E-(ABS(F/2000))
970 IF H<=0 THEN H=0
980 IF H>9935 THEN 1000
990 CALL LOCATE(#2,160-(H/(500/8)),222,#3,
    168-(H/(500/8)),222)
1000 D3=D3+D1 :: IF D3<1 THEN D3=1 ELSE IF D3>160
     THEN D3=160
1010 D4=D4+D2 :: IF D4<17 THEN D4=17 ELSE IF D4>208
     THEN D4=208
1020 CALL LOCATE(#1,D3,D4)
1030 DISPLAY AT(22,5)SIZE(6):H
1040 DISPLAY AT(22,20)SIZE(5):TIME
1050 DISPLAY AT(23,5)SIZE(5):SQR(D1^2+D2^2)*62.5
1060 DISPLAY AT(23,20)SIZE(6):E
1070 DISPLAY AT(24,5)SIZE(5):DV
1080 DISPLAY AT(24,20)SIZE(7):F/1000
1090 RETURN
1100 CALL HCHAR(22,1,32,96):: GOSUB 1160
1110 CALL CHARSET :: DISPLAY AT(22,3):"YOU CRASHED
     INTO THE HILL."
1120 IF TRIP=1 AND TIME<250 THEN FF=-22*(1000-TIME)-
     ELSE FF=FF-5000
1130 DISPLAY AT(23,3):"ALTITUDE=";H
1140 DISPLAY AT(24,3):"VELOCITY=";V
1150 GOTO 1190
1160 FOR REP=1 TO 5 :: CALL SOUND(300,110,0,110,0,
     110,0,-8,0):: CALL PATTERN(#1,99)
1170 CALL SOUND(400,110,0,110,0,220,0,-4,0):: CALL
     PATTERN(#1,100):: NEXT REP
1180 CALL CLEAR :: CALL DELSPRITE(ALL):: CALL
     CHARSET :: RETURN
1190 FOR TD=1 TO 500
1200 NEXT TD
1210 CALL CLEAR :: DISPLAY AT(20,3):"WISH TO PLAY
     AGAIN?(Y/N)"
1220 DISPLAY AT(10,3):"YOUR SCORE IS";
     (1000-TIME)+E+FF+(OPT1*OPT2*1000)
1230 ACCEPT AT(23,3)BEEP:ANS$
```

Continued on p. 87

## Mystery Words . . . from p. 42

```
210 PRINT " SOME OF THE WORDS IN THE    WORD LIST ARE NOT READILY   UNDERSTOOD BY YOUNGER CHIL- DREN.":
220 PRINT " WOULD YOU LIKE TO EXCLUDE   THESE WORDS (Y/N)?"::::::::
230 CALL KEY(0,K,S)
240 IF K=89 THEN 270
250 IF K<>78 THEN 230
260 NSET=59
270 CALL HCHAR(16,23,K)
280 GOSUB 5150
290 PRINT " THREE DRILLS ARE AVAILABLE":::TAB(5);"1) TREBLE CLEF"::TAB(5);"2) BASS CLEF"::
300 PRINT TAB(5);"3) TREBLE AND BASS":::::" YOUR CHOICE (1,2,OR 3)?":::
310 CALL KEY(0,K,S)
320 IF K<49 THEN 310
330 IF K>51 THEN 310
340 CLEF=K-48
350 CALL HCHAR(21,28,K)
360 GOSUB 5150
370 PRINT TAB(9);"INSTRUCTIONS"::"LABEL 3-9 AS A-G, 1 RED AND 0 BLUE WITH BLANK OVERLAY."
380 PRINT :"BOTH 1 AND 0 MUST BE PRESSEDAFTER TRAFFIC LIGHT IS RED."
390 PRINT :"FIRST PLAYER TO DECODE WORD PRESSES 1/0 AND USES 3-9 TO ENTER THE ANSWER."
400 PRINT ::::TAB(8);"PRESS ANY KEY"
410 CALL KEY(0,K,S)
420 IF S=0 THEN 410
430 GOSUB 5150
440 MSG$="PRESS ANY KEY TO BEGIN"
450 ROW=12
460 COL=6
470 GOSUB 4600
480 CALL KEY(0,K,S)
490 IF S=0 THEN 480
500 CALL CLEAR
510 FOR I=1 TO 8
520 CALL COLOR(I,2,16)
530 NEXT I
540 RESTORE 5110
550 GOTO 1480
560 CALL CHAR(119,"000000FFFF")
570 ON CLEF GOTO 580,710,580
580 GOSUB 4340
590 ROW=4
600 COL=12
610 MSG$="TREBLE CLEF"
620 GOSUB 4600
630 ROW=23
640 COL=10
650 F1=4
660 S1=24
670 F2=20
680 S2=39
690 GOSUB 4910
700 IF CLEF<>3 THEN 850
710 CALL CLEAR
720 R=2
730 GOSUB 4480
740 ROW=4
750 COL=13
760 MSG$="BASS CLEF"
770 GOSUB 4600
780 ROW=23
790 COL=10
800 F1=1
810 S1=21
820 F2=14
830 S2=33
*835 RESTORE 5130
840 GOSUB 4910
850 RESTORE
860 OPTION BASE 1
870 DIM SET$(59),NLINE(3),LINE(7,2,2)
880 CALL CLEAR
890 FLAG=1
900 CALL CHAR(35,"FFFFFFFFFFFFFFFF")
910 CALL CHAR(40,"")
920 CALL CHAR(96,"3C7E7E7E7E7E7E7E")
930 CALL CHAR(97,"3C3C00003C3C3C18")
940 CALL CLEAR
950 CALL COLOR(1,6,1)
960 CALL SCREEN(2)
970 CALL COLOR(2,7,7)
980 CALL COLOR(9,11,2)
990 R=24
1000 M=2
1010 MM=500
1020 FOR I=1 TO 1000
1030 NEXT I
1040 GOSUB 4710
1050 GOSUB 4840
1060 M=4
```



*Note: Line 835 was inserted as a last-minute enhancemnet to the program just prior to press time. If you have been using the automatic NUM mode, please exit it (by pressing ENTER) to type in this line. Then go back into it (NUM 840, 10) for convenience in entering the rest of the program.

Continued on p. 86

**LET COMPUTERS EARN YOUR LIVING?**

**AN OPPORTUNITY FOR YOU . . .**

- **To own your own company selling the TI 99/4A Home Computer and Software . . .**
- **As a full or part-time venture . . .**

**BENEFITS . . .**

- **Discounts on computer and software purchases**
- **Tax benefits from owning your own company**
- **Income opportunity from building an organization selling the Home Computer**

***This is a new concept, and an exciting opportunity to profit from your knowledge and experience.***

**CALL or WRITE . . .** ***COMPUTER CONNECTION***
**2301-C W. Park Row Arlington, Texas 76013**
**817/277-5106**

Chuck-A-Luck . . . from p. 25

information that is passed from one module to another is called a "variable." And deciding on what variables are needed before you sit down to write program code is just as important as deciding what modules you need. If you make a mistake in your design variables, the last phase of programming (called debugging) will take twice as long as it needs to. This is because whenever you realize that you need a new variable, you have to make coding changes in modules (that have already been coded) in order to handle them. And changing code is what destroys well-written programs!

Programs will also need variables that are used only within a module (i.e., things like loop counters), but you don't have to worry about them during your design. As long as a variable that is only used within a module has a unique name (not re-used in another module), then no problems should arise when debugging. Of course, if the variable name will be re-used in another module (which is a bad idea unless memory is tight), then it is just as important as a regular variable.

The variables that I need to communicate between my Chuck-A-Luck modules are:

The number of players
The player names
The cash each player has on hand
The amount bet by each player
The dice value on which each player bet
Value of each die

Choosing the names for these variables is equally important. A poorly chosen name is asking for trouble when you get down to writing and debugging your code. A good variable name has the following three attributes:

It is long enough to say what it is and what it's for.
It is short enough so as not to slow down the program.
It does not look too similar to any other variable.

For the *Chuck-A-Luck* game, I'll use
NO-PLAYERS, PLAYER-NAME,
PLAYER-CASH, PLAYER-BET,
PLAYER-DICE and DICE-VALUE
as my variable names. And, I won't re-use variables that are used within a module.

Now my design is finally complete and I'm ready to start coding. I have done everything that I could to insure that the program will do its job, and am ready for the sixth step in creating good program—planning the code. But that's the subject of my next article. For now, just remember that the first rule of good programming is PLAN, PLAN, PLAN!

**THE MICRO HOUSE**

Complete TI-99/4 Line

✶ Discount Prices ✶

| | |
|---|---|
| Extended BASIC | $81 |
| TI Invaders | 33 |
| Carwars | 33 |
| Tombstone City | 33 |
| Mail List Merge | 15 |

**Add $5 for Shipping & Handling**

*For complete price list write:*

THE MICRO HOUSE
527 Simonet Street
Green Bay, WI 54301
Tel. (414) 432-2871

TI-99/4 Software

- **Astronomy Programs**
- **Matrix Routines**
- **Linear Equation Solver**
- **Fourier Transform**
- **Music Analysis/Synthesis**
- **Curve Fitting Routines**
- **Hyperbolic Functions**
- **Two-Dimensional FFT**
- **Data Fitter**

**Programming in TI BASIC Extended BASIC and UCSD Pascal**

**Programs also Available for the TRS80 in Level 2 BASIC**

*For further information write:*
**Eastbench Software Products**
**1290 Cliffside Drive**
**Logan, Utah 84321**
**Tel: (801) 753-1084**

Psychometric
*An eight part mental health inventory*
**Measures which psychological areas (below) may need further testing:**

- **PERSONALITY FACTORS**
- **LIFESTYLE STRESS**
- **FEAR**
- **ANXIETY**
- **MARRIAGE, FAMILY RELATIONSHIPS**
- **JOB OR OCCUPATIONAL DEPRESSION**
- **NON-FAMILY PERSONAL RELATIONSHIPS**

Offered only as an aid to individual self-awareness; this program does not provide any psychiatric treatment, is not a substitute for professional psychological counseling, nor is it intended as such.
Available only on cassette, runs on **99/4** or **99/4A**
**Program with instructions for interpretation . . . . . . . . . . . . . . . . $8.95**
Send cash, check, or money order. Shipped prepaid anywhere in the U. S. California residents add 6% state sales tax.

Pablo Diablo,
The legendary evil kiteflier
P. O. Box 4863
Santa Clara, CA 95054
SOURCE ID TCV774

# LOGO TIMES

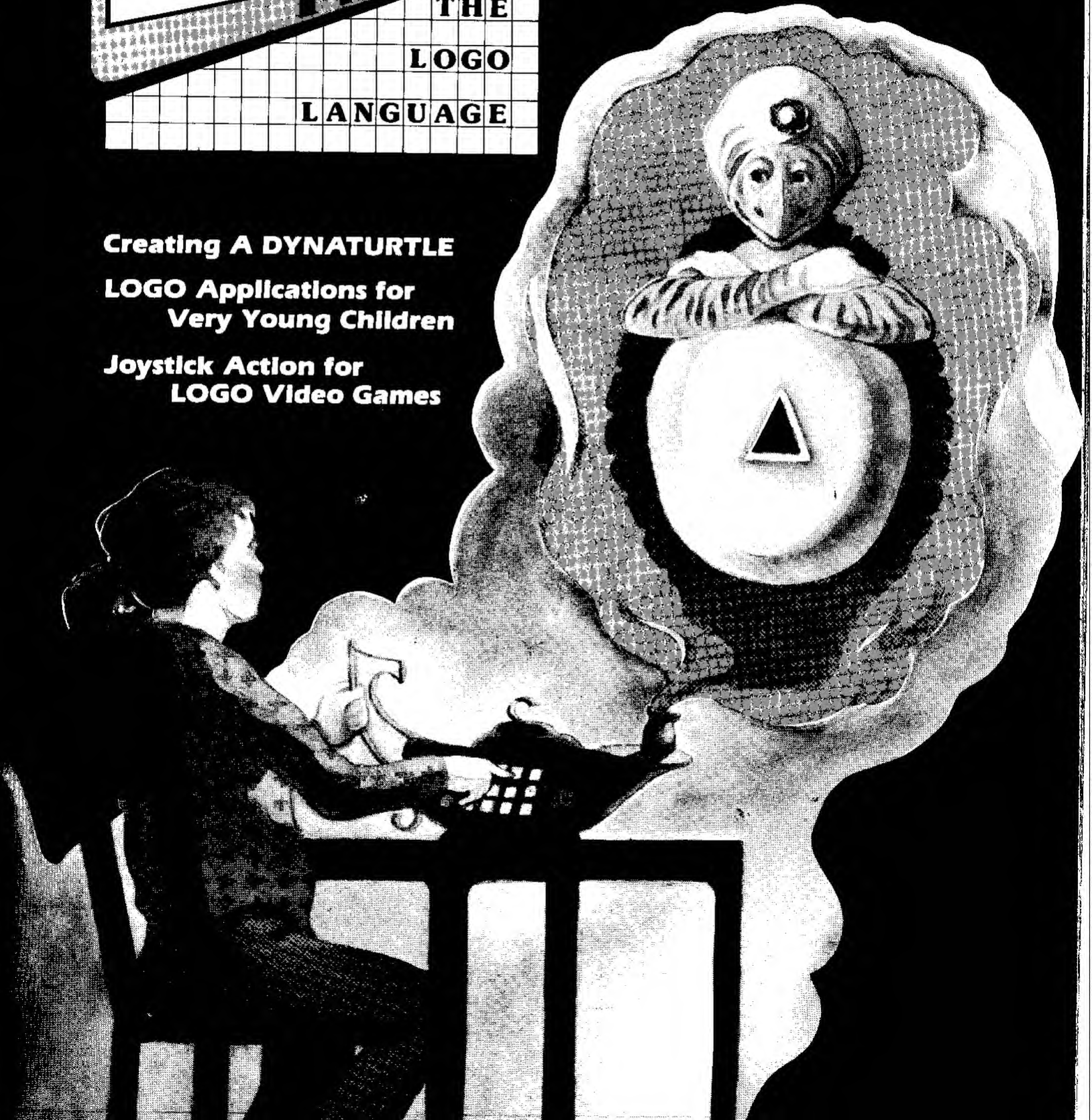
THE MAGAZINE OF THE LOGO LANGUAGE

## The LOGO Genie: Your Wish Is His Command

Creating A DYNATURTLE

LOGO Applications for Very Young Children

Joystick Action for LOGO Video Games



## Introduction

*LOGO Times* is an information resource for anyone interested in participating in the creation of their own *personal* language – one that will easily allow them to communicate with a computer in a totally new audiovisual realm of applied imagination, exploration, and self-discovery. The articles on these pages concern the use of the new TI LOGO language, but readers, however, do *not* need any additional software or equipment (or even a computer) to understand and learn from the material presented here.

If readers want to actually *experience* a TI LOGO environment, they will need either a TI-99/4 or TI-99/4A computer, the Expansion Memory peripheral, and the TI LOGO Command Module. A disk drive, although convenient to have, is *not* required; a user's work may alternately be saved on cassette tape, printed out on the TI Thermal Printer, or hand copied into a notebook (for later re-keyboarding).

In each issue, one or more of the articles may reference or build upon the topics discussed in a previous article. It is therefore recommended that for maximum benefit and understanding, new readers obtain the appropriate back issues of *99'er Magazine* in which the *LOGO Times* articles are contained.

### Notice

*LOGO Times* is actively soliciting articles. Manuscripts should be typed double-spaced, and accompanied by a cassette tape or disk if containing any lengthy procedures or graphics.

Send all materials to:

LOGO Times Editorial Dept.
99'er Magazine
2715 Terrace View Drive
Eugene, OR 97405

All mail directed to the Letters-to-the Editor column (*Letters on LOGO*) will be published in accordance with the conditions set forth on *99'er Magazine's* contents pages.

**Our Contributing Editors**

Henry Gorman, Jr.
Department of Psychology
Austin College
Box 1584
Sherman, TX 75090

Roger B. Kirchner
Department of Mathematics
Carleton College
Northfield, MN 55057

James H. Muller
Young Peoples' LOGO Association
1208 Hillsdale Drive
Richardson, TX 75081

# LOGO'S POWE

By Roger B. Kirchner

## PART 1. An Overview of the Language Structure and Syntax

Logo was developed by Seymour Papert and his associates at the MIT Artificial Intelligence Laboratory in order to study the way people might learn in a computer-rich environment. It was designed to be a language so simple to use that a person could manipulate objects or concepts by just thinking about what he or she wanted to accomplish, and not having to worry about programming. Such a language might stimulate a person to explore, to learn, and grow.

The idea was to provide certain *primitive* commands and operations that could be combined to form more *complex* commands and operations. These more complex ones could then be used exactly like the primitive ones. Thus it would be possible to construct a single command to accomplish anything that could be accomplished using the primitive concepts. Additionally, *recursion*—whereby a command could call and activate itself—was to be allowed.

LOGO is a relative of LISP, the list processing language used in artificial intelligence. LOGO and LISP share the capability of manipulating numbers, words (character strings without a space), and lists. A *list* is a recursively defined object: It is an ordered set of objects, each of which may be a number, a word, or a previously defined list. In LOGO, a *procedure* is represented by a list; there are commands to access a list that represents any procedure, and to define *new* procedures from lists which might be the result of some manipulation. Furthermore, a procedure may have inputs and may have an output, and is activated by specifying its name (a word) followed by its inputs (which may be numbers, words, or lists). Defined procedures as well as primitive commands and operations all have exactly the *same* syntax. This is why LOGO is so simple to use. Its power comes from its list processing capabilities.

I hope that the description given so far has made it apparent that LOGO is *not* just for children. It is much more than FORWARD 20 RIGHT 90, but it also can be used in elementary ways. LOGO is a language for all people who want to learn and expand their capacities.

### The LOGO Turtle

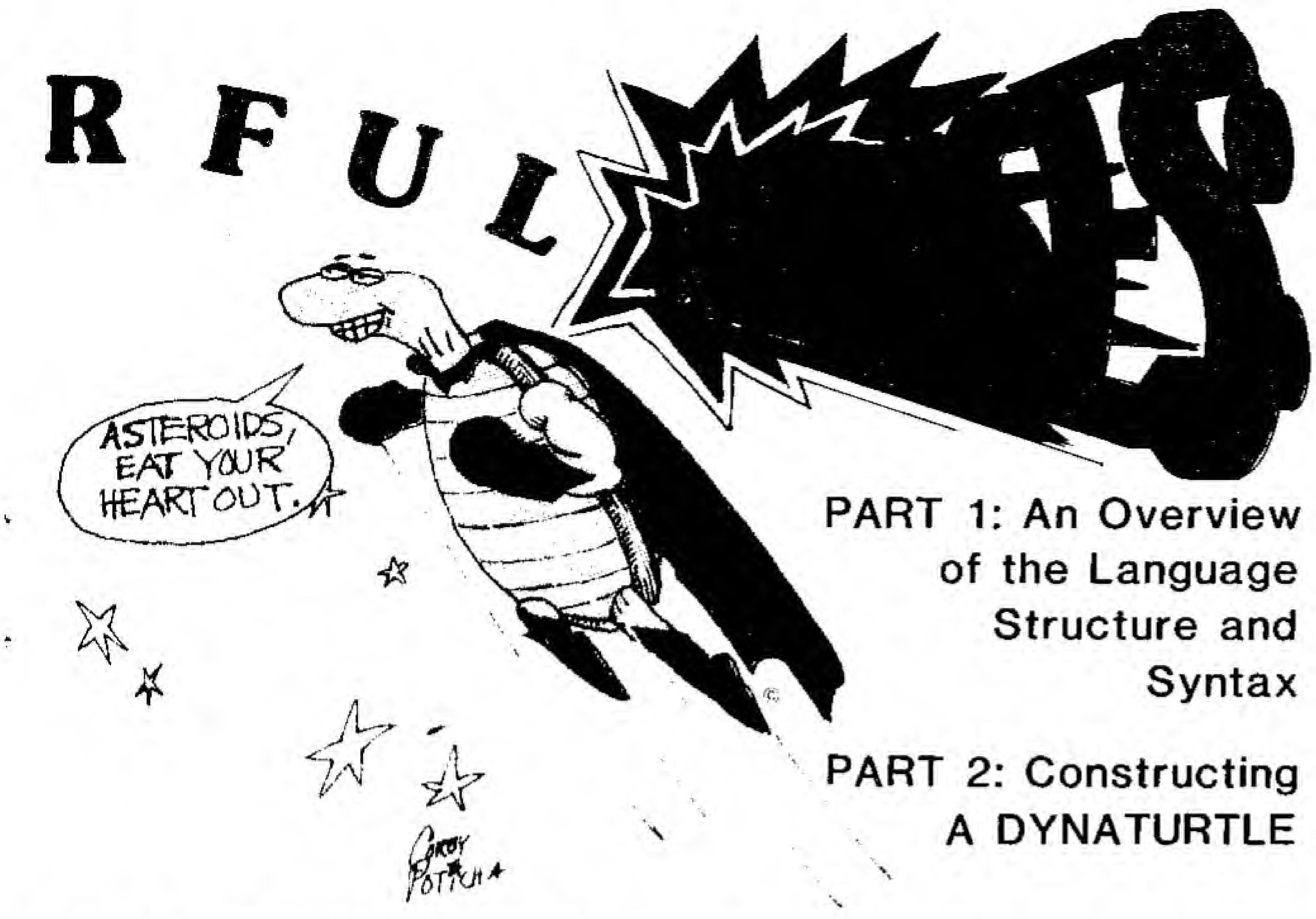
The first experiments with LOGO were with junior high school students who could appreciate manipulation of words. Then the Turtle was created whose movements could be understood by very young people.

The Turtle was originally a robot that could be commanded to move about the floor. It had a pen which could be either up or down. In an experiment at the University of Pittsburg Learning Center several years ago, one young person used LOGO to command the floor turtle to draw an alphabet of large letters. He also taught it to act like an airplane, and "fly" between cities on a large map. The plane had the possibility of going out of control, with the turtle going into a spiral and spinning on the floor. The turtle is now usually a small triangle on a terminal screen, but it can still do such things—albeit, on a smaller scale.

At the youngest levels, LOGO is being used to teach a feeling for distances and angles. At levels up through the college level it is being used to advance a new subject in mathematics called "Turtle Geometry." Some interesting theoretical results have been discovered. (A wealth of examples and exercises is contained in *Turtle Geometry* by Abelson and diSessa, where procedures are expressed in a language almost exactly the same as LOGO.) Recursive designs such as snowflake curves, space filling curves, and trees are applications of LOGO's power.

### TI LOGO

TI LOGO is being marketed as a language for children, and it was a pleasant surprize to discover that TI LOGO has all of the list processing capabilities built into it. All the recursive designs presented in *Turtle Geometry* can be drawn.



## PART 1: An Overview of the Language Structure and Syntax

## PART 2: Constructing A DYNATURTLE

(The TI Turtle is, however, limited to 192 different 8x8 pixel character positions. Thus, if a figure is very dense, it can't be very large.)

The process of discovering the power of TI LOGO is not made easy by the documentation that comes with it. Many of the commands needed for manipulating all but the simplest lists are not documented.

At this point, it may be helpful to briefly describe just what is available to a person who sits down to use TI LOGO: The TI Turtle is an object that lives on a coordinate screen with horizontal coordinates from −119 to +120 and vertical coordinates from −46 to +97. The bottom six lines of the screen are used for text. The turtle can be assigned a position, and "knows" where it is. It can be assigned a heading (from 0 to 360 as the points of a compass) and knows its heading. Its heading can be changed by a given angle, and it can be moved either in the direction or opposite to the direction or opposite to the direction of its heading a given amount. It can make a dot at any position. The pen can be down, up, or in "reverse" modes, and it can draw in any of 15 colors.

Unique to the TI version of LOGO are *sprites*—objects familiar to those with TI EXTENDED BASIC. There are 32 sprites (numbered 0 to 31) with each assigned to a 16x16 pixel shape. Users may design and store 26 of these and can direct any collection of sprites to simultaneously assume an attribute such as shape, color, position, heading, speed, or velocity. The commands which control the turtle act similarly on the sprites. Motion is controlled by assigning a speed (in the current direction) or a velocity (horizontal and vertical components). Not only can attributes be assigned, but they can also be obtained as the output of operations since a sprite always knows its own number, shape number, color number, position (on the full screen), heading, speed, and velocity.

Papert has described "Velocity Turtles" (they can be assigned velocities) and "Acceleration Turtles" (their velocities can be incremented). Sprites can be both. Using sprites we can even simulate Papert's "Dynaturtle"—an acceleration turtle which does not change direction when it is rotated, but changes velocity only by accelerating in the direction it is facing, thus obeying Newton's laws of motion. A dynaturtle therefore behaves like the ship in the popular *Asteroids* arcade game. The example procedures that follow this article will demonstrate a dynaturtle which can have the force of its "thruster" changed, and which can simulate an environment with friction.

TI LOGO also has 256 tiles (numbered 0 to 255) that can be given arbitrary 8x8 pixel designs. We can assign tiles foreground and background colors, and position them anywhere on the 24x32 character screen or on the current print line. Console characters are tiles—the tile number being the ASCII code of the character. (Note: The Turtle records its trace using tiles, so simultaneous use of the Turtle and nonprinting characters is limited.)

### Numbers, Words, and Lists

A *number* in TI LOGO is an integer from −32,768 to 32,767. Numbers can be added, subtracted, multiplied and divided (integer quotient), calculations being modulo 32,768. The restriction to integer arithmetic is a definite limitation, but the limitation is not serious for most applications.

A *word* is a character string without a space. A feature of LOGO distinguishing it from other programming languages such as BASIC or Pascal is the capability of using a word *simultaneously* as (1) the name of a command or procedure, (2) a variable, and (3) data. For example, if the word X is to be used as the name of an action, X itself is used. When an object has been assigned to X, the object is denoted :X. And the word X as data is denoted "X. Suppose that X has not been defined as an action and has not been assigned a value. LOGO will respond to X with **TELL ME HOW TO X**, to :X with **:X HAS NO VALUE**, and to "X with **TELL ME WHAT TO DO WITH X**.

A word can be assigned any kind of data—i.e., a number, word, or list as a value. This also distinguishes LOGO from BASIC or Pascal where the data type of a variable must be specified in advance. As a bizarre example, note that **MAKE "MAKE "MAKE** and **MAKE "MAKE [MAKE]** assign to **MAKE** first the word **MAKE** and then the list whose single member is the word **MAKE**.

A *list* is the most powerful data object in TI LOGO and is denoted by a left bracket followed by its members, then a right bracket. Examples of lists are [ ] , the null list, **[HOW NOW BROWN COW]** a list of words, and **[REPEAT 4 [FORWARD 20 RIGHT 90]]** a list whose members are a word, a number, and another list.

### Data Manipulation in LOGO

Commands which are powerful in manipulating data include the following: **FIRST(F), LAST, BUTFIRST(BF), BUTLAST (BL), SENTENCE (SE), FPUT, LPUT, NUMBER? , WORD? , THING?, THING, WORD, MAKE, RUN, TEXT, DEFINE.** The last three are used to execute a list of commands to access the list which represents a procedure, and to define a procedure represented by a given list. These are powerful commands, but to be able to make use of them it is necessary to be able to construct lists whose members themselves are lists. The following key (undocumented) commands, **FPUT** and **LPUT** are helpful here:

**FPUT** object list—outputs a list whose first member is object, and whose following members are the members of list.

**LPUT** object list—outputs a list whose last member is object and whose all but last members are the members of list.

If object is a word or a number, the results of these commands are the same as **SENTENCE object list** and **SENTENCE list object**, respectively. But if object is a list, **FPUT object list** adds object to the beginning of list while **SENTENCE object list** adds the *members* of object to the beginning of list. This is a crucial difference, making possible the construction of arbitrarily complicated lists. The other commands in the above list which are undocumented are as follows:

**NUMBER?** object—returns **TRUE** if object is a number, and **FALSE** otherwise.

**WORD?** object—returns **TRUE** if object is a word, and **FALSE** otherwise.

**THING?** "name—returns **TRUE** if name has been assigned a value, and **FALSE** otherwise.

**THING** "name—returns the object which has been assigned to name, if name has a value.

**WORD** word1 word2—returns the word formed by concatenating word1 and word2. (Compare with **SENTENCE**, below.)

**SENTENCE** wordorlist1 wordorlist2 (a documented command—returns a list determined by the inputs. If an input is a word, that word is put in the list. If an input is a list, its members are included in the list.

Some of the undocumented commands were found by accident and by studying the documentation for MIT LOGO. Others were known to Jim Muller, president of the Young Peoples' LOGO Association (YPLA). We encourage readers to share other discoveries with us.

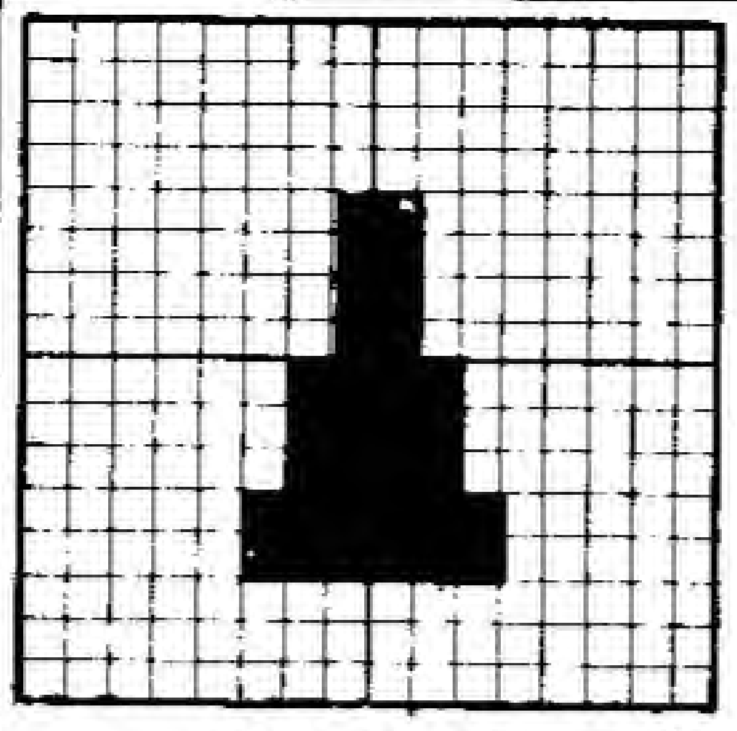
### A Calculating Example

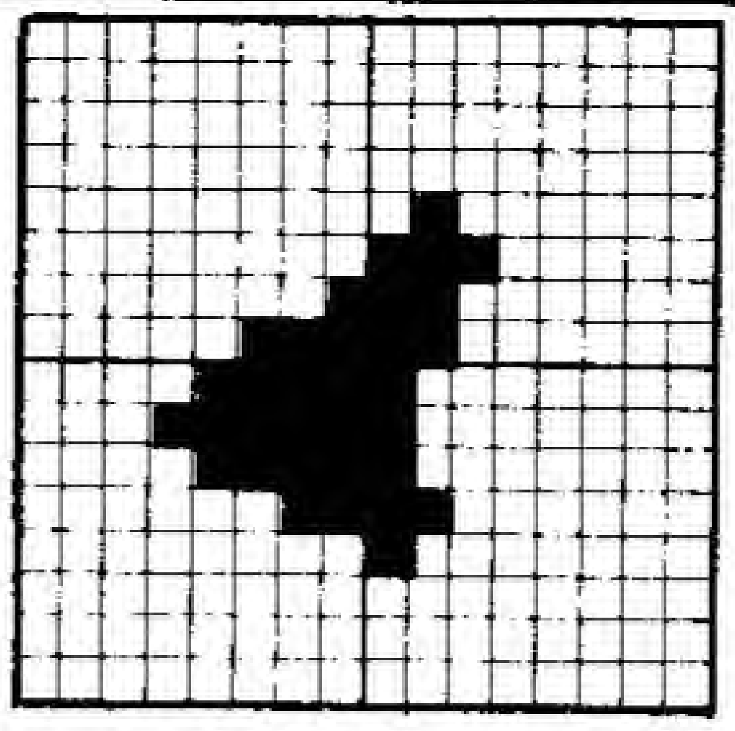
As a simple example, consider the problem of teaching LOGO to act like a calculator. If one enters 2+3, the response is **TELL ME WHAT TO DO WITH 2+3**. Here, desired output is 5, which is the result of executing **PRINT 2 + 3**. The problem is solved by using **SENTENCE** to form the list **[PRINT 2+3]** and then using **RUN** to execute the list. A solution is the following:

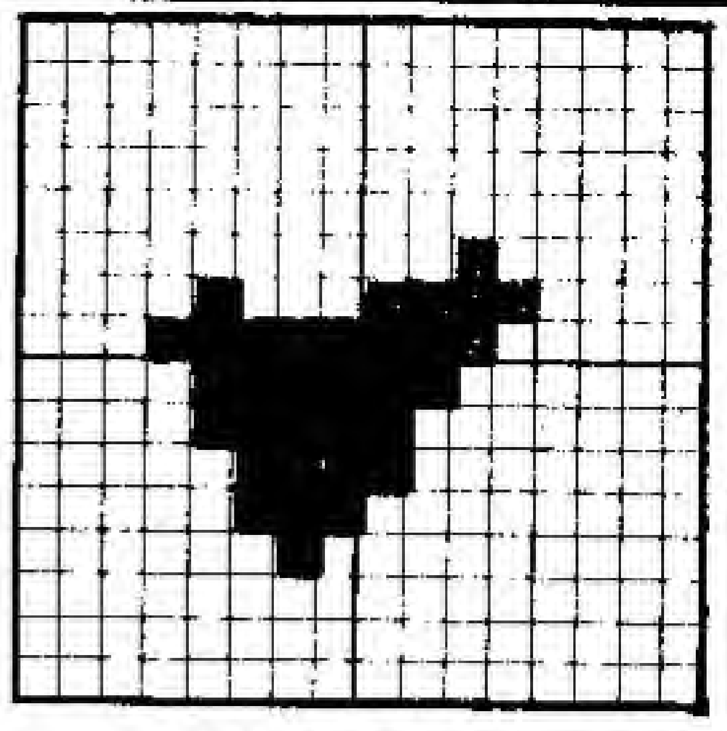
```
TO CALCULATE
MAKE "Y READLINE
TEST :Y=[ ]
IFF RUN SENTENCE "PRINT :Y
  CALCULATE
END
```

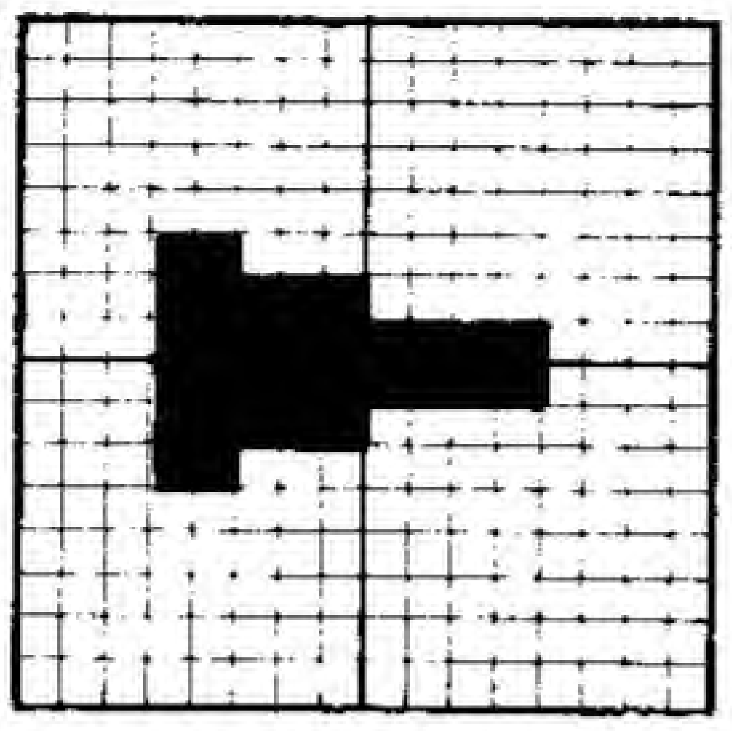
After entering **CALCULATE**, the computer accepts arithmetic expressions and prints out the resulting value until just ENTER is pressed. The recursion then "unwinds" and the procedure stops.

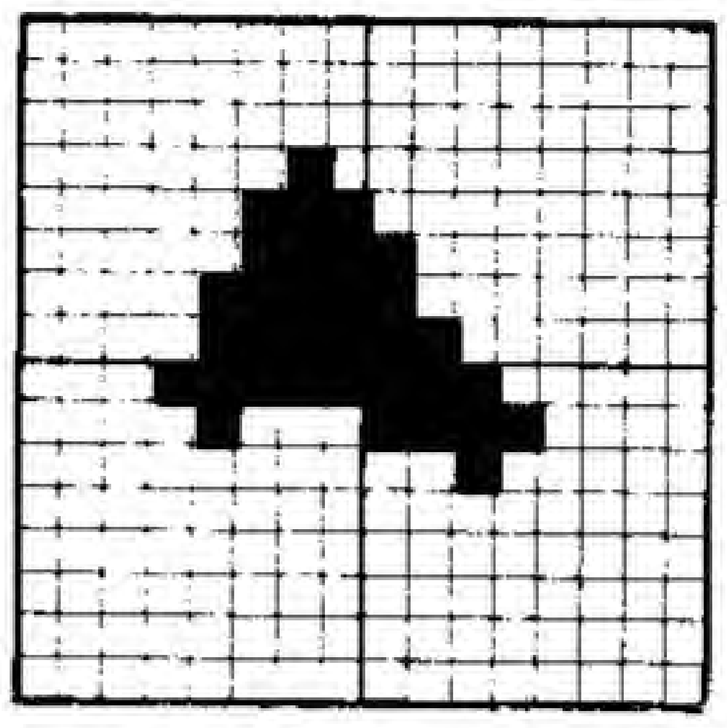
The power of a list processing language such as TI LOGO becomes apparent the more you use it. Yet for learning, all of these advanced capabilities *don't* have to be utilized. This is what makes the language so versatile—its built-in power that is accessible on demand. And it is this versatility that allows teachers to tailor LOGO for special applications, and reassures all students that with LOGO there is always more to learn . . .

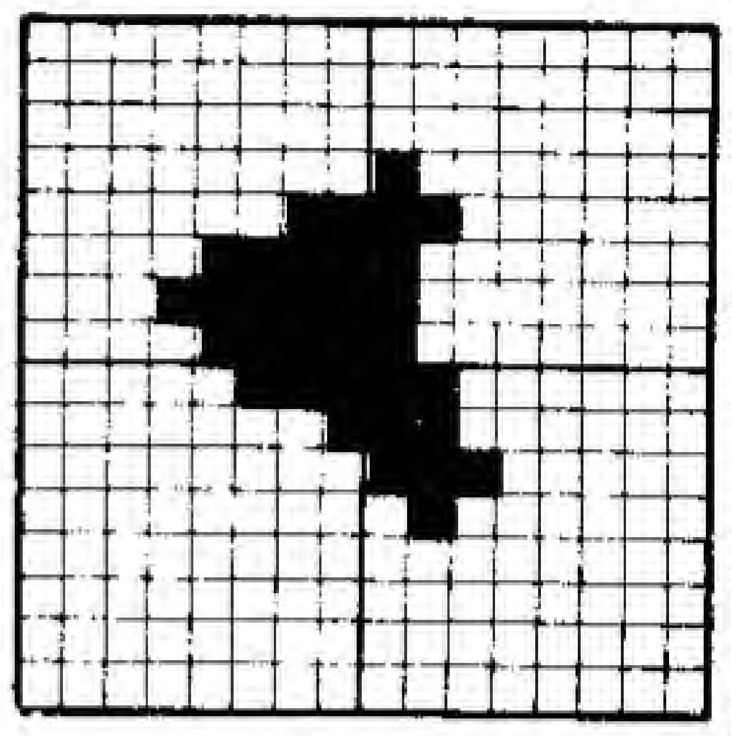

SHAPE 10 △

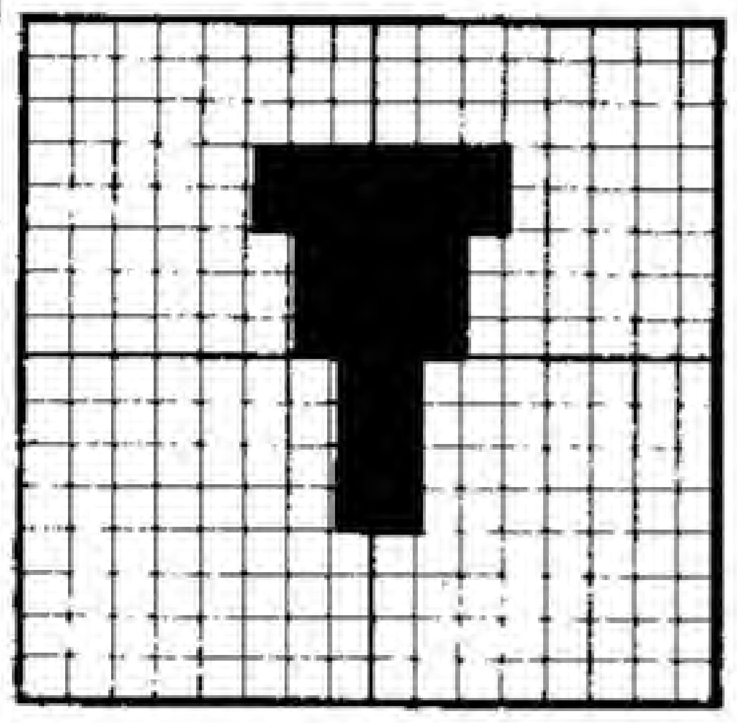

SHAPE 11 ◁

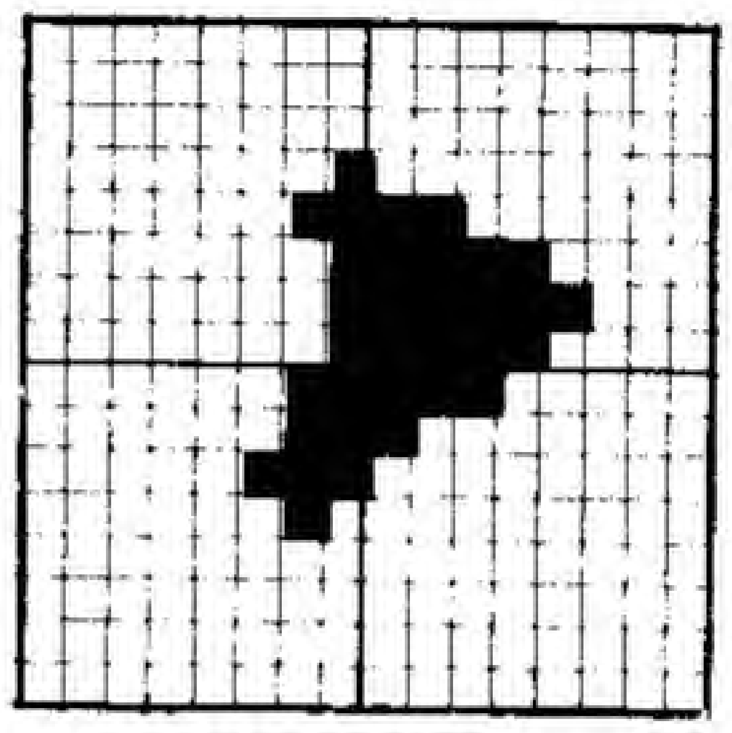

SHAPE 12 ▽

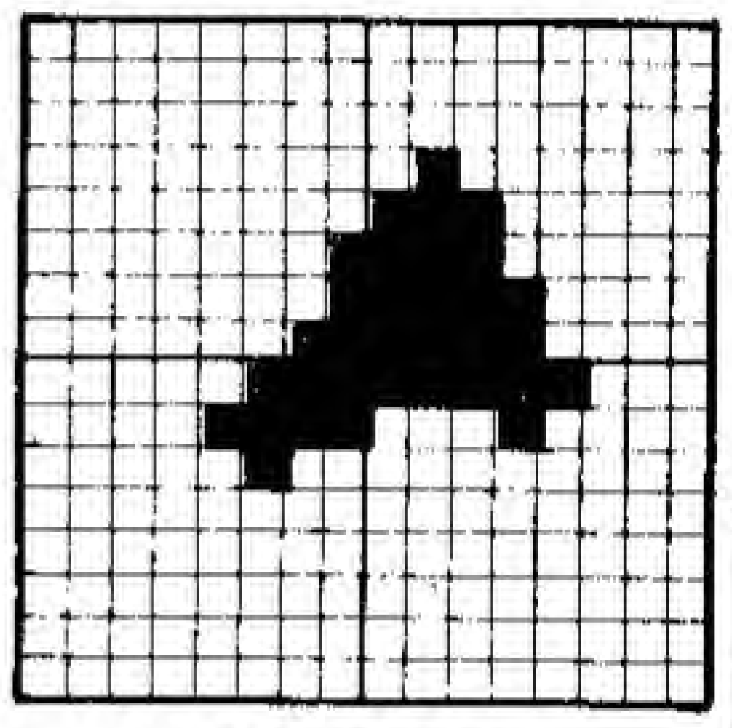

SHAPE 13 ▷

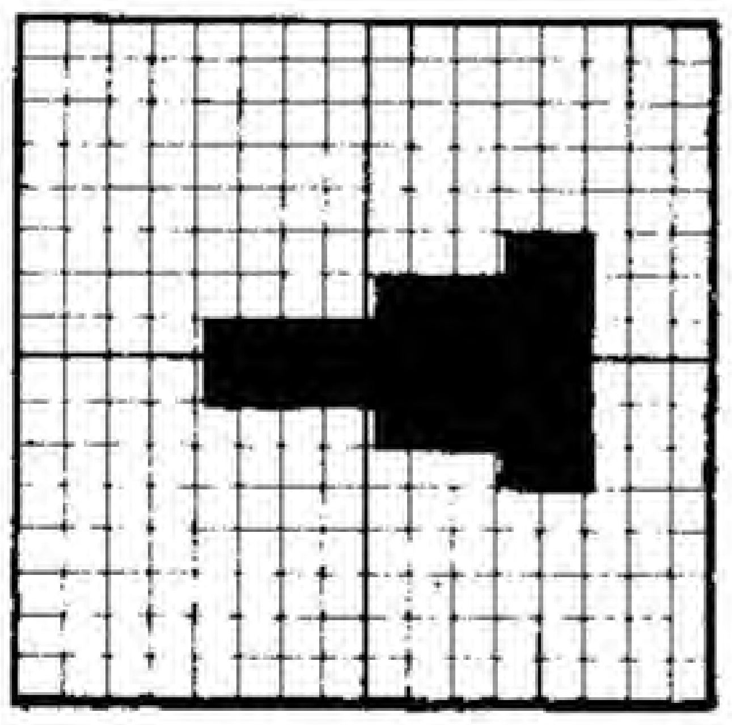

SHAPE 14 △

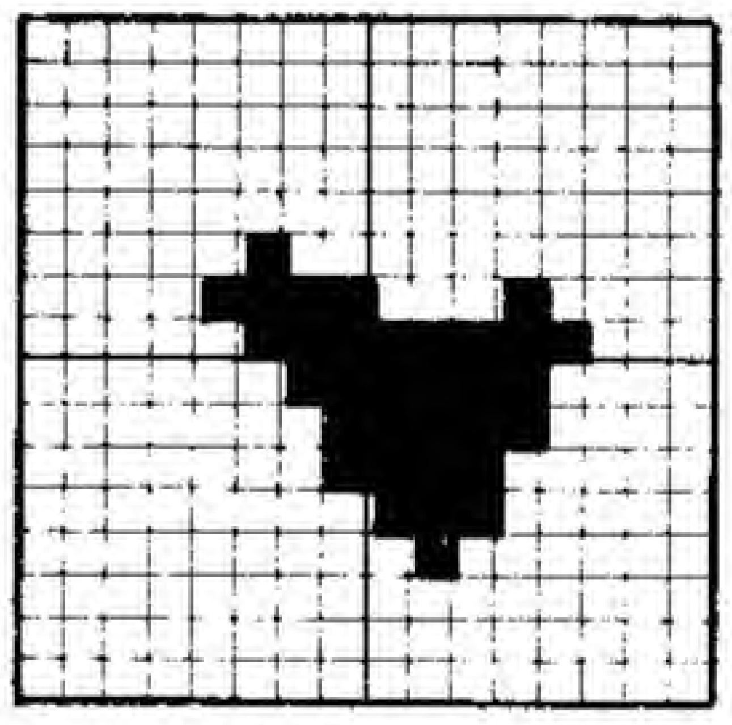

SHAPE 15 ◁

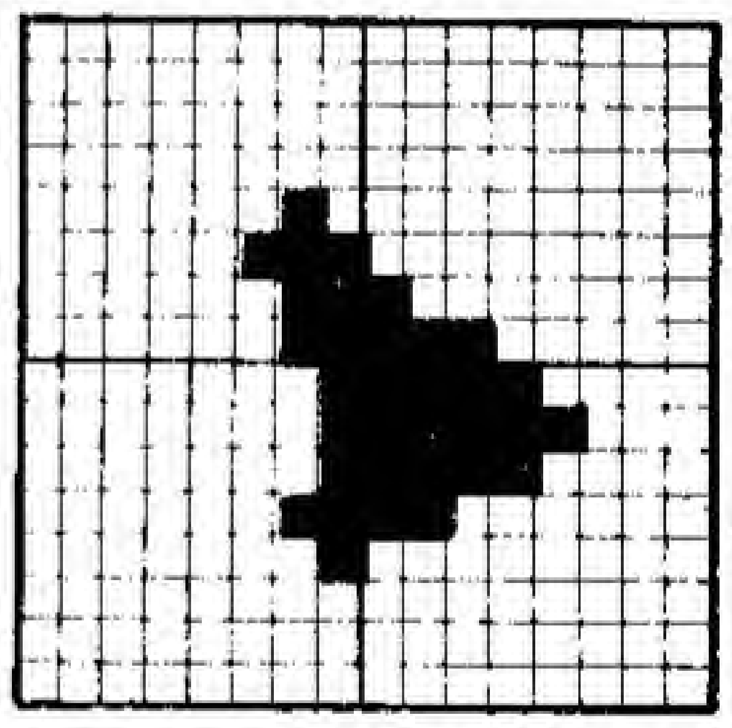

SHAPE 16 ▽


SHAPE 17 ▷


SHAPE 18 △


SHAPE 19 ◁


SHAPE 20 ▽


SHAPE 21 ▷

## PART 2: Constructing A DYNATURTLE

The instructions for using the dynaturtle are obtained by typing **HELP**. The dynaturtle itself is activated by typing **DYNATURTLE**. The procedure starts out drawing a circle and displaying a white dynaturtle. Touching the **E** key causes a "thruster" to impart motion to the dynaturtle with speed 3. Each touch of the **E** key adds a velocity with magnitude 3 to the dynaturtle. Touching **S** or **D** makes the dynaturtle face 30 degrees left or 30 degrees right from its former heading. Touching **E** always imparts an additional velocity with magnitude 3 in the direction the dynaturtle is heading. Velocities add like vectors. If the dynaturtle is not facing in the direction of its motion, the force of the thruster will cause it to head in a direction intermediate between its heading and direction, exactly as if it were a rocket in space obeying Newton's laws.

Touching **F** will turn friction on. In this state, the dynaturtle will be sluggish and come quickly to a stop after each kick. It will therefore be necessary to increase the force of the thruster. To do this, touch **K**. You can then enter a number, say 10 or 20, and touch ENTER. The dynaturtle will now be given an increase in velocity with magnitude 10 or 20 with each touch of **E**. Touching **F** again will turn friction off. You will find the dynaturtle now very difficult to control. Touch **K** again and readjust the thrust.

When friction is off, the dynaturtle is seen to act just like the ship in the *Asteroids* arcade game. When friction is on, it behaves as if it were riding on a rough surface—appearing to skid as you direct it around the circle.

## Description of Procedures

**DYNATURTLE** activates the procedures **INITIALIZE, SETDYNATURTLE**, and **CONTROL**.

**INITIALIZE** draws a circle and initializes the thruster (sprite 0).

**SETDYNATURTLE** positions the dynaturtle and gives it its initial shape (shape 10). The secret of the dynaturtle's turning capability is that the twelve shapes (shape 10 through shape 21) contain designs for the dynaturtle—each rotated 30 degrees from the preceding.

**CONTROL** is the main "loop." Friction is always checked to see if it is on. If it is on, **CHECKFRICTION** decreases the dynaturtle's speed. If one of the control keys is pressed, the action is taken and control branches to lable **A**. This procedure keeps running until **Q** is touched.

**KICK** reads the velocity of sprite 0 which is always kept heading in the direction the dynaturtle is facing. This velocity is then added to the velocity of sprite 1 which carries the shape of the dynaturtle.

**ROTRIGHT** adds 30 degrees to **H** which maintains the heading of the dynaturtle, and causes sprite 1 to carry the shape with next highest number, unless that number is larger than 21. If sprite 1 is carrying shape 21, it assumes shape 10. In this way, the dynaturtle appears to be rotating to the right by 30 degrees.

**ROTLEFT** is similar to **ROTRIGHT** but gives the effect of rotating the dynaturtle to the left.

**SETFRICTION** simply makes the value of the word **FRICTION?** true if it is false, and false if it is true.

**SETKICK** gets a number from the console and assigns it as the speed for sprite 0. The velocity for sprite 0 (x- and y- coordinates) is used to impart an acceleration to sprite 1. Note the command **SS FIRST READLINE**. The primitive **READLINE** outputs a list, and **SS** requires a number for input. The desired number is the first (only) member of the list entered.

```
PROCEDURES
----------

TO SETKICK
TYPE [FORCE OF KICK? ]
TELL 0
SS FIRST READLINE
END

TO SETFRICTION
TEST :FRICTION?
IFT MAKE "FRICTION? "FALSE PRINT
 [FRICTION OFF ]
IFF MAKE "FRICTION? "TRUE PRINT
[FRICTION ON ]
END

TO ROTRIGHT
TELL 1
MAKE "H :H + 30
IF SHAPE = 21 THEN CARRY 10 ELSE
 CARRY SHAPE + 1
END

TO ROTLEFT
TELL 1
MAKE "H :H - 30
IF SHAPE = 10 THEN CARRY 21 ELSE
 CARRY SHAPE - 1
END

TO CHECKFRICTION
IF :FRICTION? THEN TELL 1 IF SPE
ED > 0 THEN SS SPEED - 1
END

TO CIRCLE SIDE
REPEAT 36 [FD :SIDE LT 10 ]
END

TO CONTROL
A:
CHECKFRICTION
TEST RC?
IFF GO "A
MAKE "X RC
IF :X = "E THEN KICK
IF :X = "S THEN ROTLEFT
IF :X = "D THEN ROTRIGHT
IF :X = "F THEN SETFRICTION
IF :X = "K THEN SETKICK
IF :X = "Q THEN STOP
GO "A
END

TO SETDYNATURTLE
TELL 1
SX 50 SY 8
SS 0 SH 0 CARRY 10
SC :WHITE
END

TO INITIALIZE
MAKE "H 0 ; HEADING OF DYNAT
TELL TURTLE ; DRAW CIRCLE TO GO
AROUND
CS HT
SX 50 CIRCLE 8
TELL 0 ; INITIALIZE THRUSTER
SS 3
MAKE "FRICTION? "FALSE
END

TO KICK
TELL 0
SH :H
MAKE "DVX XVEL MAKE "DVY YVEL
TELL 1
SV XVEL + :DVX YVEL + :DVY
END

TO DYNATURTLE
INITIALIZE
SETDYNATURTLE
CONTROL
END

TO HELP
NOTURTLE CS
PRINT [THE DYNATURTLE IS AN OBJECT ]
PRINT [WHICH OBEYS NEWTON'S LAWS ]
PRINT [OF MOTION. ]
PRINT [ ]
PRINT [IT LIVES ON A SURFACE WHICH ]
PRINT [CAN BE SMOOTH OR ROUGH. ]

PRINT [ ]
PRINT [CONTROLS: ( TOUCH KEY ) ]

PRINT [ ]
PRINT [E: GET KICK FROM THRUSTER ]
PRINT [S: TURN LEFT 30 DEG ]
PRINT [D: TURN RIGHT 30 DEG ]
PRINT [F: TURN FRICTION ON / OFF ]
PRINT [K: SET THRUSTER KICK ]
PRINT [Q: QUIT ]
PRINT [ ]
PRINT [TRY TO GO AROUND CIRCLE. ]
PRINT [ ]
PRINT [TYPE "DYNATURTLE" ]
END
```

## Letters on LOGO

**AN OPEN LETTER TO EDITORS, EDUCATORS, PARENTS AND INTERESTED LOGOPHILES:**

I have just returned from the Microcomputers in Education Conference at Arizona State University in Phoenix, Arizona. While there, I was privileged to present the concept of the Young Peoples' LOGO Association along with some of the work that our young people have accomplished.

While explaining LOGO to educators and computer scientists, I was struck by the variety of misconceptions that exist about this beautiful language and the genius of Seymour Papert. How limited is the view of some people of Dr. Papert and the language that he and his MIT colleagues have developed . . .

From recent issues of *InfoWorld* come the following quotations:

*"I like the LOGO stuff, but when (Papert) writes about what children can do—like drawing on the screen—I could see spending 5, 10, 20, maybe even 100 hours doing that; but at some point I would think to myself, 'Well, I've done that; I'm not interested in learning how to draw fancier pictures. I'd rather work with the piano or cello."*

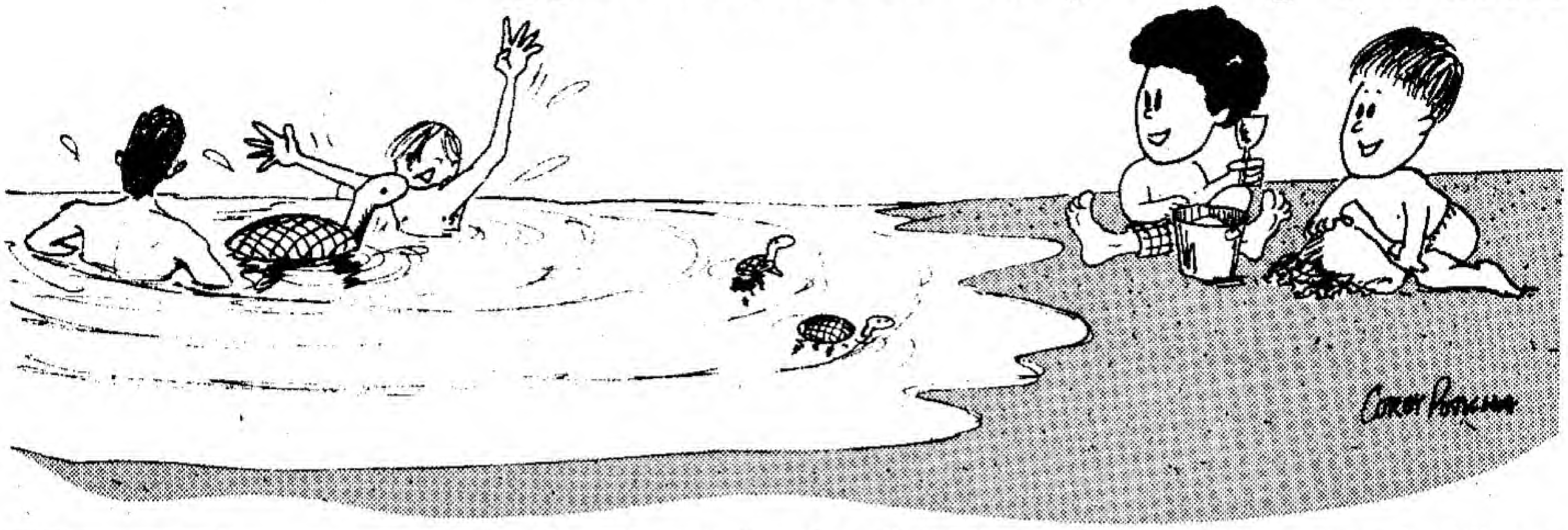
*"LOGO is a wonderful language, but to get LOGO processes into the computer, kids have to type the words TELL, CARRY, SPEED, DIRECTION and so forth, over and over. Just like they do with BASIC. Great! . . . But, do they ever get a chance to experience 'playing' the LOGO language as a musician experiences playing a piano? At a minimum, can the repetitive aspects of the LOGO language be relegated to single keystrokes on a modified (relabeled) keyboard."*

First of all, to compare LOGO to BASIC is ludicrous. Secondly, both of these contributors to *InfoWorld* (and others to other publications) have missed the most important contribution that LOGO makes to the learning process. Plain and simply, you start with what you know—a simple, direct conversational language. You analyze a situation, break it down into its component elements, and systematically develop the procedure to solve the problem. If you want the computer to do something, you teach it how. TELL, CARRY, FORWARD, BACK SETCOLOR, SETSPEED are nothing more than primitive commands—tools that the user need enter only once to develop his own language, to develop the single or multiple keystroke tool that he can use to paint imaginative pictures, to write beautiful prose, or to create beautiful music. The primitives are tools, nothing more or less. They are the pencils and paper used to tap the imagination.

The real genius of LOGO is that young people can start with their everyday vocabulary—the form of expression with which they are most familiar—and move into a totally new audio visual realm where they can explore, experience, discover, and learn the fundamentals of geometry, math, text manipulation and a host of other opportunities. It doesn't matter if this is 1882, 1982, or 2082. Unless the functioning of the human brain changes drastically, the fundamentals of learning will remain the same—you break a problem down into the component elements with which your experience will allow you to work, and then you use these elements to procedurally discover new knowledge. In essence, you start with the known and move to the unknown.

Continued on p. 74

# EXT-E-N-D-I-N-G LOGO: Applications for Very Young Children



By Henry Gorman, Jr.

Seymour Papert designed LOGO to have a low threshold so that even young children could benefit from it. Unfortunately, the present technical requirement that LOGO input and output be *text-bound* limits LOGO to children who can read and type.

It is apparent that learning in a LOGO environment offers greater potential for pre-verbal children than for verbal humans. This is simply because there is so much more for them to learn. But it is equally apparent that the reading/typing prerequisite is an artificial barrier to that same environment. The ultimate solution, a computer which can comprehend *and* generate speech is not yet available. (Although such programs as TI text-to-speech can do a reasonable job of talking). Still, there are ways LOGO can be adapted to children who are only able to recognize alphabetic characters or typewriter keyboard symbols.

Even before LOGO was implemented on the DEC LSI/11 or the TI-99/4, people in the MIT LOGO lab worked on simplified LOGO systems. One approach yielded a special LOGO input device which translated symbol cards[1] inserted into slots through a light scanner into ASCII code. Although prototypes of the "slot machine" card reader worked well enough, the idea was never commercially developed. A second approach was followed by Bob Lawler, a graduate student at the time, who wanted his two children to be able to use LOGO. He wrote a number of excellent LOGO programs which allowed *very* young children to draw with the turtle, to play shoot-out games with the turtle, or to design elaborate turtle pictures. Lawler's programs were written for a large, mainframe computer version of LOGO, but his ideas are amenable to TI LOGO. In fact because of the excellent color graphics of TI LOGO, his ideas may more effectively involve children when set up on the TI-99/4A. The essence of his programs was to simplify access to turtle geometry by simplifying and combining commands. One simplification which allows pupils to move the turtle forward or backward a fixed amount, or to turn the turtle left or right a fixed amount by pressing any of the four "arrow" keys is:

```
TO DRAW
TELL TURTLE
SIMPLIFY
END
```

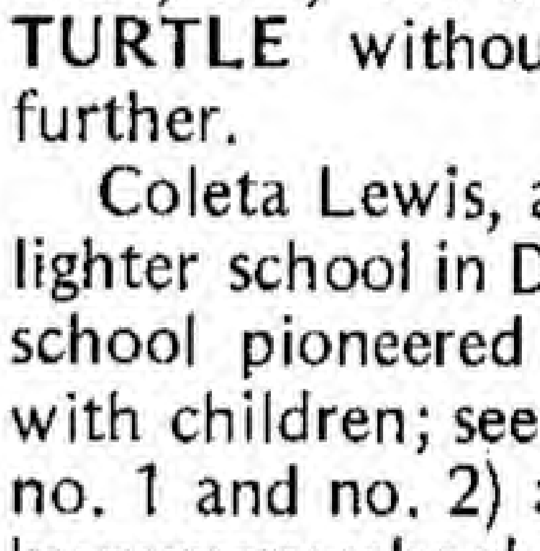
```
TO SIMPLIFY
IF :ANSWER = "D RIGHT 30
CALL RC "ANSWER
IF :ANSWER = "E FORWARD 10
IF :ANSWER = "X BACK 10
IF :ANSWER = "S LEFT 30
SIMPLIFY
END
```

With slightly more sophisticated children, or as children work with **DRAW**, we can add other commands by merely inserting them into **SIMPLIFY**. For example:

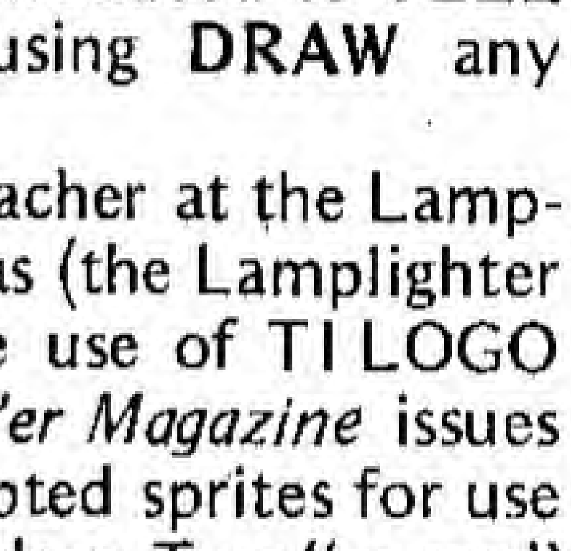
```
TO SIMPLIFY
IF :ANSWER = "C CLEARSCREEN
IF :ANSWER = "Q STOP
IF :ANSWER = "0 PENERASE
IF :ANSWER = "1 PENDOWN SETCOLOR
 1
IF :ANSWER = "4 PENDOWN SETCOLOR
 4
IF :ANSWER = "U PENUP
```

Then, as students master the fuller set of **DRAW** commands and learn the idosyncracies of QWERTY typewriter code, they can be introduced to **TELL TURTLE** without using **DRAW** any further.

Coleta Lewis, a teacher at the Lamplighter school in Dallas (the Lamplighter school pioneered the use of TI LOGO with children; see *99'er Magazine* issues no. 1 and no. 2) adapted sprites for use by nursery school children. Two "games" her children played allowed them to either move a garage around the screen, move a car around the screen, and vary the colors of each separately, or to construct a face and then color in the parts of the face. Programming sprites for very young children is not much more difficult than **DRAW**. As one example of a sprite game for youngsters, consider a game of blocks. It is fairly easy to create a universe of blocks with simplified sprite commands. First, it is necessary to make up some "blocks" using **MAKESHAPE**. A circle (shape 4) and a square (shape5) already exist. A good set of blocks ought to have a triangle.



SHAPE 4

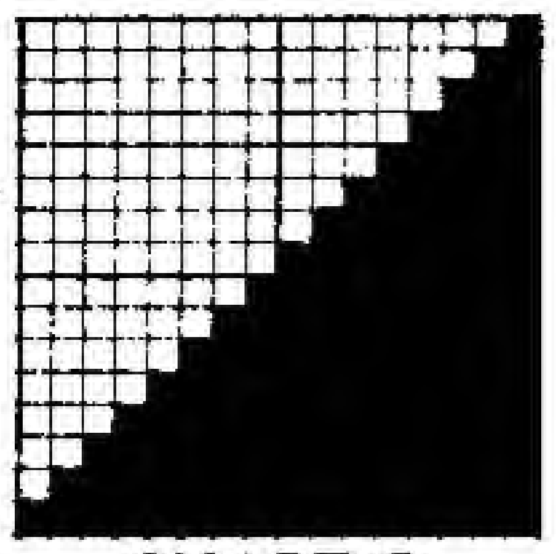


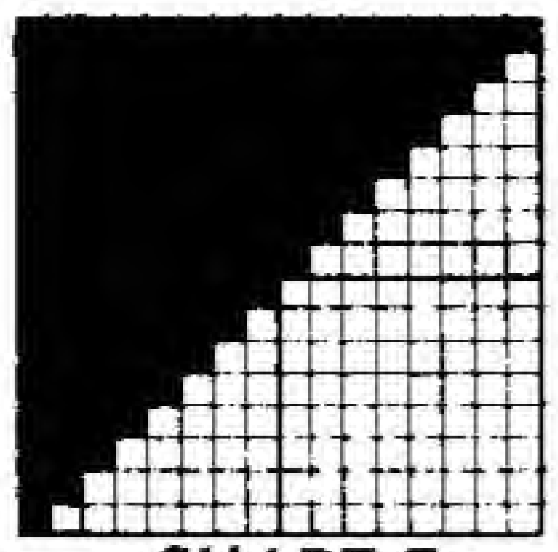
SHAPE 5

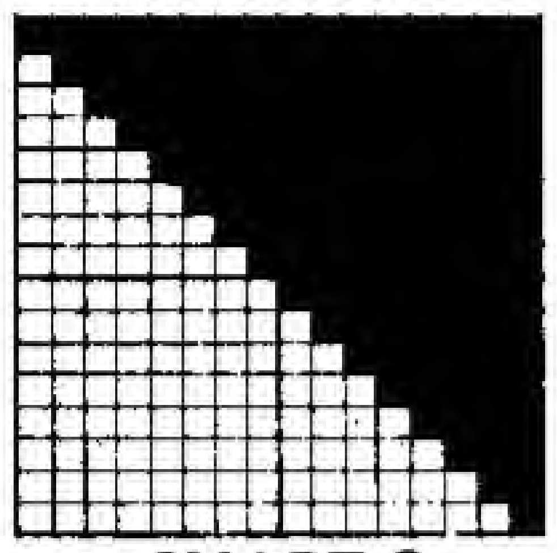
For this, however, it would be better to build triangles with four different orientations since the shapes of sprites cannot be rotated.

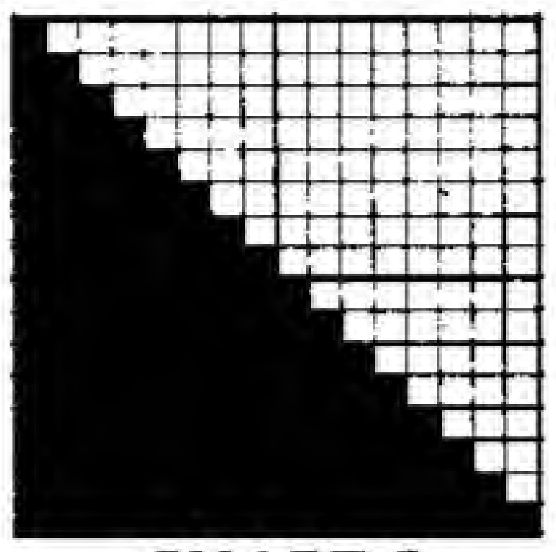
These could be assigned shape number 6, 7, 8, and 9. Also, a good block set should have a rectangle. To show the

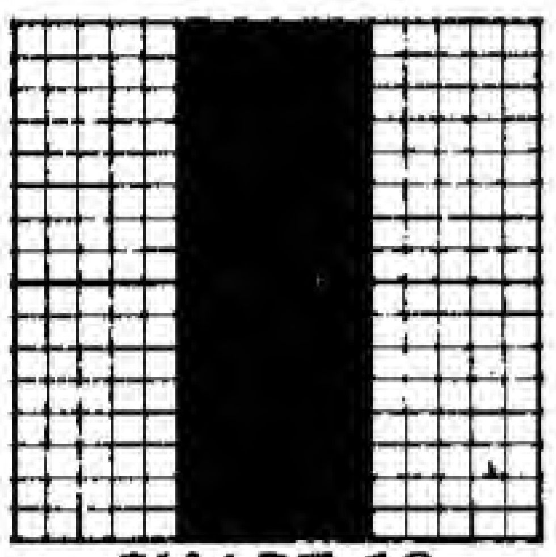
[1] The cards carry labels like ⌐→ for RIGHT 90, ↑ for FORWARD 10, as well as symbols for recursion and sub-procedure calls.

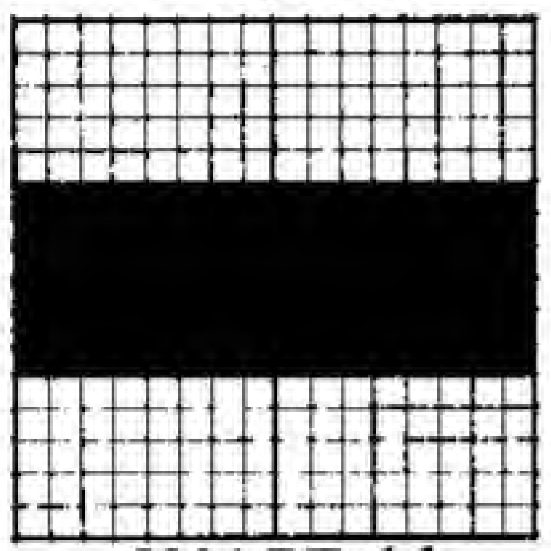

SHAPE 6


SHAPE 7


SHAPE 8


SHAPE 9


SHAPE 10


SHAPE 11

two orientations, shape numbers 10 and 11 could be designed as shown.

Other shapes could be easily added. Children should be able to color the blocks, move the blocks around, change the shape of any of the blocks, and to bring more blocks onto the screen. Also, when a child begins working with one of the blocks, he or she should be able to identify which one it is. (One way this can be accomplished is to have them briefly flash colors. All of this is implemented in the following programs:

```
TO BLOCKS
            ; this program cleans up the screen
              and gets BUILD started
TELL :ALL
SETCOLOR 0 CARRY 4
HOME
CLEARSCREEN
TELL 0
BUILD
END

TO BUILD  ; this program takes a single key
            typed as input and uses it to
            change sprite shape, color, num-
            ber, location
CALL RC "ANSWER
IF :ANSWER = "N NEXT
IF :ANSWER = "B BACKONE
IF :ANSWER = "C SETCOLOR COLOR +
 1
IF :ANSWER = "E SETHEADING 0 FOR
WARD 10   ; this moves the sprite 10 steps in
            the direction of the arrow
IF :ANSWER = "X SETHEADING 180 F
ORWARD 10; this also moves 10 steps in the
            direction of the arrow
IF :ANSWER = "S SETHEADING 270 F
ORWARD 10; moves the sprite 10 steps in the
            direction of the arrow
IF :ANSWER = "D SETHEADING 90 FO
RWARD 10  ; moves the sprite 10 steps in the
            direction of the arrow
IF :ANSWER = "Q STOP
IF :ANSWER = "F FORM
            ; this allows the student to stop
BUILD       ; this makes the program recursive
              so the game will continue
END

TO NEXT   ; this recruits the next sprite
TEST YOURNUMBER = 31
IFT PRINT [THERE ARE NO MORES SP
RITES LEFT. ] STOP
IFF TELL ( YOURNUMBER + 1 ) FLAS
H CARRY 4 STOP
END

TO BACKONE
            ; this moves back to the previous-
              ly recruited sprite
TEST YOURNUMBER = 0
IFT PRINT [THIS IS THE FIRST SPR
ITE. ] STOP
IFF TELL ( YOURNUMBER - 1 ) FLAS
H STOP
END

TO FORM   ; this allows the child to change
            shapes
TEST SHAPE = 11
            ; with the block set made so far,
              11 is the last block and 4 the
              first, a larger set could be taken
              care of by changing this number
IFT CARRY 4
            ; this command starts off with the
              first block in the set
IFF CARRY ( SHAPE + 1 )
            ; this command changes the shape
              to the next shape in the set
END

TO FLASH  ; this command causes either
            FLASH0 or FLASH1 so that the
            sprite will flash briefly to identify
            the current sprite
CALL COLOR "START
IF :START = 0 FLASH0 STOP
FLASH1 :START STOP
            ; this command causes the sprite
              to flash and finish flashing with
              its original color only when the
              color isn't color 0 (invisible).
END

TO FLASH0
REPEAT 5 [SETCOLOR 1 WAIT 20 SET
COLOR 0 WAIT 20 ]
END

TO FLASH1 :START
REPEAT 5 [SETCOLOR 0 WAIT 20 SET
COLOR :START WAIT 20 ]
END
```

In the last issue of *99'er Magazine* my LOGO column carried a poetry program. The **POET** program used the subprocedures **COUNT, LINE, LINES, VERSE, SELECT, NUMB,** and **SPACE** to generate lines of poetry which were a random number of words per line with a random number of spaces between words (where random is a quasi-randomly selected digit between 0 and 9). the "work" of getting a randomly selected word from a list (which is input into the program) was accomplished by the subprocedures **LINE** and **VERSE.**

At the end of the article, I suggested an exercise of converting **POET** into a program which either produced rhyming verse, blank verse, or a finite number of lines of verse. One way to modify **POET** to produce rhymed verse is to give it two different lists—one of words for the interior words of each line of verse, and the other rhyming words for the last word in each line. Then the program could be changed so that only rhyming words were placed in end positions.

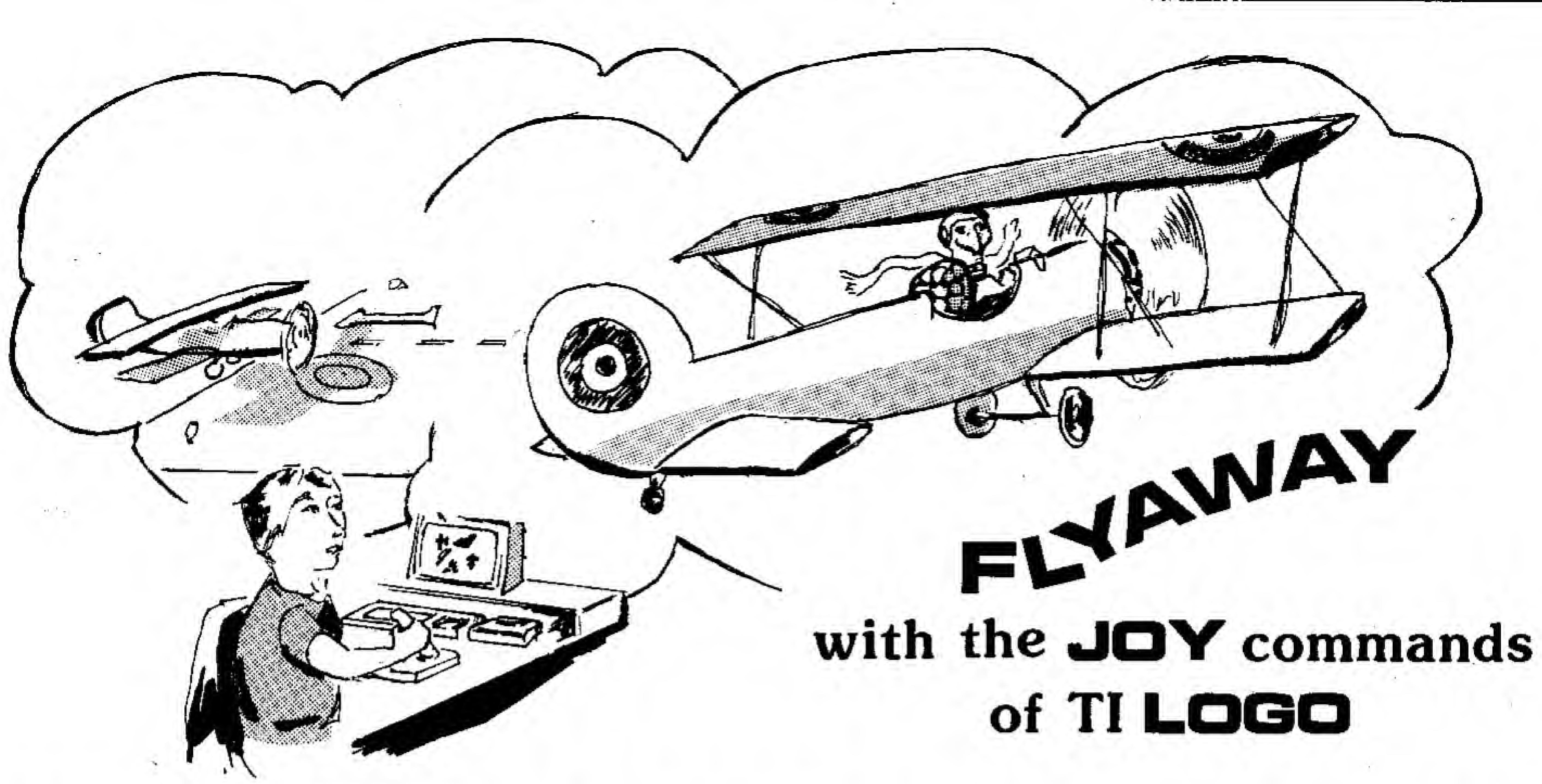
```
TO POET :LIST :RHYMES
            ; the second list must now be given
              the program
CALL COUNT :LIST "LENGTH
CALL COUNT :RHYMES "LENGTHR
            ; this is necessary to find out how
              many rhyming words there are
LINES :LIST :RHYMES
END

TO LINES :LIST :RHYMES
            ; LINES must be changed to ac-
              comodate two lists
LINE :LIST :RHYMES
LINES :LIST :RHYMES
END

TO LINE :LIST :RHYMES
            ; finally, LINE the workhorse
              of POET must be changed to put
              only rhymes at the end of each line
REPEAT RANDOM [SPACE VERSE :LIST
 ]
PRINT SELECT ( NUMB :LENGTHR ) :
RHYMES    ; this puts a rhyme at the end
END

TO SPACE
FC 32
END

TO COUNT :LIST
IF :LIST = [ ] OUTPUT 0
OUTPUT ( COUNT BUTFIRST :LIST )
+ 1
END

TO NUMB :LENGTH
CALL RANDOM "N
TEST BOTH :N > 0 :N < ( :LENGTH
+ 1 )
IFT OUTPUT :N
IFF OUTPUT NUMB :LENGTH
END

TO SELECT :N :LIST
IF :N = 1 OUTPUT FIRST :LIST
OUTPUT SELECT :N - 1 BUTFIRST :L
IST
END

TO VERSE :LIST
TYPE SELECT ( NUMB :LENGTH ) :LI
ST
END
```

You probably recognize that the problem of generating rhyming verse is one form of the problem of teaching the computer to write text which follows a specified rule (in this case each line must rhyme). The more general application of rules to text is nothing less than grammar. One of the grade school pupils in the Brookline project wrote a text-rule program like POET which generated random sentences. After she saw the effects of changing parts of speech she exclaimed enthusiastically that she now understood what a noun was.

POET can also be quickly adapted to a sentence generator which young people can play with to make grammar meaningful.

```
TO SENTENCES
PRINT [TYPE A LIST OF ARTICLES A
ND THENPRESS ENTER. ]
CALL READLINE "ART
PRINT [TYPE A LIST OF NOUNS AND
THEN PRESS ENTER. ]
CALL READLINE "NOUNS
PRINT [TYPE A LIST OF ADJECTIVES
 AND PRESS ENTER. ]
```

Continued on p. 77



# FLYAWAY with the JOY commands of TI LOGO

By James H. Muller
with Roger Kirchner

Pushing the stick forward, the aircraft begins to roll. You then gradually pick up speed and start moving down the runway. When reaching takeoff speed, you move the stick again, and suddenly you're airborne. Now you have control of the skies—to fly high or low, do loops and other maneuvers, and then land. But be careful with your speed! You don't want to stall and crash.

This isn't flight training or a simulator. It is a TI LOGO procedure that gives you the opportunity to fly by keyboard or joystick. It uses either the arrow keys or the **JOY 1** and **JOY 2** commands. The **JOY** commands return one of nine values depending on the position of the joystick, thus opening a wide range of possibilities for interactive games and other activities.

At first, it might appear that the nine values have little relationship to each other. You'll note that the three and the seven were omitted. However, the pattern of the values is quite interesting.

$$\begin{tikzcd}
2 & 6 & 10 \\
1 & 5 \arrow[l] \arrow[r] \arrow[u] \arrow[d] \arrow[ul] \arrow[ur] \arrow[dl] \arrow[dr] & 9 \\
0 & 4 & 8
\end{tikzcd}$$

Moving from left to right in each row, you'll note that each digit is four more than the previous digit—i.e., 2 + 4 = 6, 6 + 4 = 10, etc. Moving from bottom to top in each column, observe that each digit is one more than the previous one.

These patterns begin to suggest why three and seven were omitted from the values assigned. However, to make the logic behind these patterns even more graphic, let's convert them to binary numbers.

$$\begin{tikzcd}
0010 & 0110 & 1010 \\
0001 & 0101 \arrow[l] \arrow[r] \arrow[u] \arrow[d] \arrow[ul] \arrow[ur] \arrow[dl] \arrow[dr] & 1001 \\
0000 & 0100 & 1000
\end{tikzcd}$$

Now look at the first two digits in each column. You'll note that they are the same, representing from left to right, 0, 1, and 2. Also, if you look at the last two digits in each row, you'll note that they are also the same. Moving from bottom to top, they also represent 0, 1, and 2. So what we really have here is a distinctive coordinate system with real meaning, rather than what might first be perceived as a random placement of values.

$$\begin{tikzcd}
(0,2) & (1,2) & (2,2) \\
(0,1) & (1,1) \arrow[l] \arrow[r] \arrow[u] \arrow[d] \arrow[ul] \arrow[ur] \arrow[dl] \arrow[dr] & (2,1) \\
(0,0) & (0,1) & (2,0)
\end{tikzcd}$$

Let X and Y be used to name these coordinates. The coordinates for the joysticks can be assigned with the command:

```
MAKE "Y (JOY 1)/4
MAKE "X (JOY 1) – 4 * :Y
```

Now let's put these **JOY** commands to work in **FLYAWAY**, a procedure developed by Roger Kirchner, a fellow YPLA member [see his related article in this issue—Ed.]. This is a procedure for one or two players that tests each player's ability to take off and safely land an airplane on the runway shown on the screen. Control of the plane is by the direction keys on the keyboard, or by joysticks.

The joystick commands are incorporated in the procedure, **STICK S**. Push the stick forward and the aircraft increases its speed.

```
IF :X = 6 THEN FASTER
```

Pull the stick back and the aircraft slows down:

```
IF :X = 4 THEN SLOWER
```

To minimize the chance for error, direction commands are accessed by merely moving the joystick to the left or right. It does not matter whether you hit position 0, 1, or 2, the aircraft will turn left.

```
IF :X<4 THEN TURNLEFT
IF :X>6 THEN TURNRIGHT
```

Of course, it would be possible to add additional maneuvers using each of the nine joystick positions. This would require a much more sensitive touch to the joystick but that could also add to the challenge of the flight.

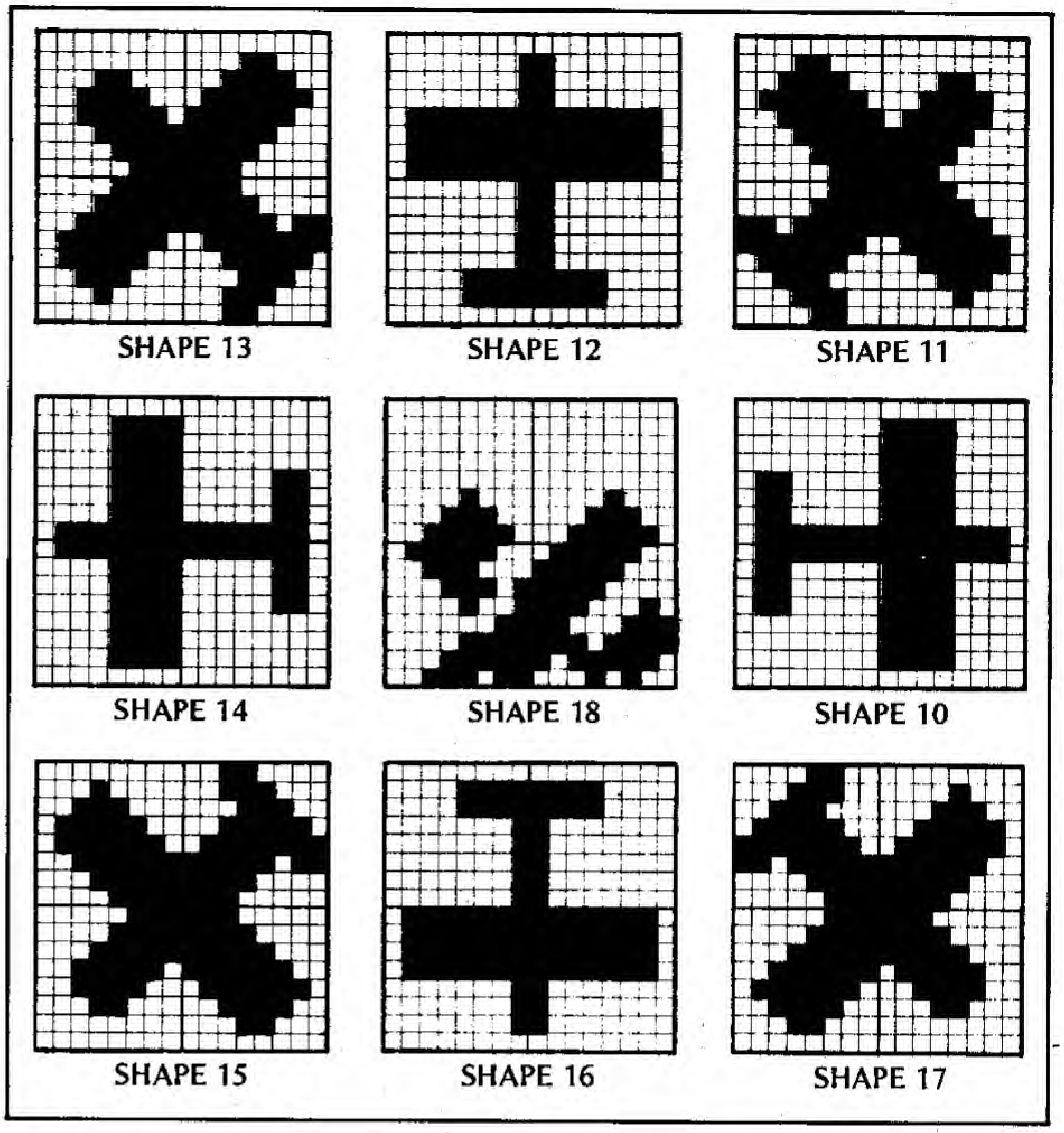
With each turn of the aircraft, a new shape is called to show that new position. These range from #10 through #18. The first shape, #10, is similiar to the Plane shape in TI LOGO. The next shape shows the aircraft at a 45° angle. The other shapes depict the plane in a 90° angle, 135° angle, at 180°, 225°, 270°, and 315°. Shape #18 depicts the crash.

Control of the aircraft in the air is done by the **CONTROLJOY** and the **CHECK P** procedures. **CHECK P** monitors the speed and "altitude" of the aircraft to test for the **CRASH** parameters.

**FLYAWAY** is an excellent graphic program that begins to tap the power of the LOGO language. It goes quite a bit further than merely drawing pictures with the computer. This is an important point to realize. But unfortunately, not all educators do: In a November issue of *Infoworld*, one educator stated that although he can understand young people enjoying maybe 50 to 100 hours of drawing pictures with LOGO, he would imagine they would then tire of the language and move on to the other things.

I ask you: Do chess players ever tire of chess? Do chess masters ever really feel that they have mastered the game? I think not. The moves of chess can be easily learned by primary grade youngsters, but entire *lifetimes* are spent learning the game. Certainly the graphic capabilities and the speed of TI LOGO are spectacular. Indeed, they tend to overshadow the other attributes of the language. Where that happens, it is most unfortunate because LOGO offers a young person so much more than just graphics—much more than BASIC and some other high level languages.

For example, look closely at BASIC; it uses a finite number of commands that must be strung together in statements that tend to hide the operation of the program. Were the operation of BASIC programs easily discerned, TRACE would be unnecessary. LOGO, on the other hand, is a virtually unlimited language. If the command doesn't exist, use your imagination and create it! This is the marvellous challenge of TI LOGO—using your imagination and creativity to discover the *real* potential of the computer. Way back in the "dark ages" before microcomputers, Albert Einstein expressed a truth that is especially relevant to today's computer learning environments: "Imagination is more powerful than knowledge." Undoubtedly, Einstein would have approved of TI LOGO . . .

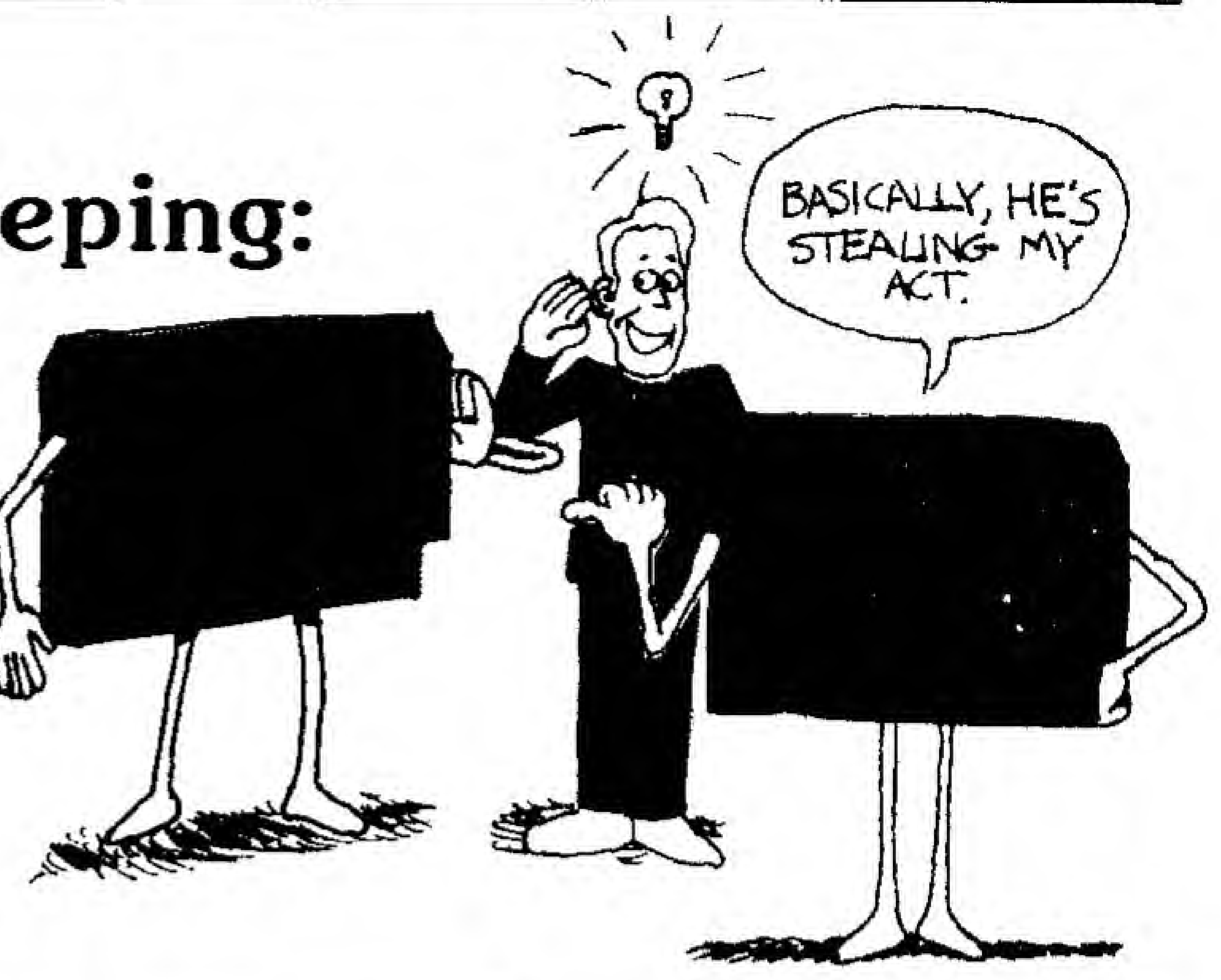


```
PROCEDURES
----------
TO FLYAWAY
SETUP
RUNWAY
SETPLANE 1 SETPLANE 2
TEST :MODE = "J
IFF CONTROL
IFT CONTROLJOY
END

TO SETUP
CS
VANISH
PRINT [INSTRUCTIONS ( Y / N ) ?
]
IF RC = "Y THEN HELP MAKE "X RC
CS
PRINT [NAME OF BLUE PILOT? ]
MAKE "PILOT1 READLINE
PRINT [NAME OF RED PILOT? ]
MAKE "PILOT2 READLINE
PRINT [ ]
PRINT [FLYING WITH KEY CONTROLS
OR JOYSTICKS ( K / J ) ? ]
MAKE "MODE F READLINE
MAKE "XS ( - 100 )
MAKE "Y1S ( - 40 )
MAKE "Y2S ( - 60 )
MAKE "UPOFF ( - 30 )
MAKE "DOWNOFF ( - 72 )
CS
END

TO RUNWAY
TELL TILE 96 SC [2 15 ]
TELL TILE 104 SC :GREEN
MAKE "TILES [104 104 100 100 96
100 100 104 104 ]
MAKE "ROW 15
A:
IF :TILES = [ ] THEN STOP
MAKE "T F :TILES
MAKE "COL 0
REPEAT 32 [PT :T :COL :ROW MAKE
"COL :COL + 1 ]
MAKE "TILES BF :TILES
MAKE "ROW :ROW + 1
GO "A
END

TO CONTROL
MAKE "X RC
IF :X = "E THEN TELL 1 FASTER
IF :X = "I THEN TELL 2 FASTER
IF :X = "X THEN TELL 1 SLOWER
IF :X = "M THEN TELL 2 SLOWER
IF :X = "S THEN TELL 1 TURNLEFT
IF :X = "J THEN TELL 2 TURNLEFT
IF :X = "D THEN TELL 1 TURNRIGHT

IF :X = "K THEN TELL 2 TURNRIGHT

IF :X = "Q THEN STOP
CHECK 1 CHECK 2
CONTROL
END

TO CONTROLJOY
STICK 1 CHECK 1
STICK 2 CHECK 2
IF RC? THEN STOP
CONTROLJOY
END

TO VANISH
TELL :ALL HOME SH 0 SS 0 CARRY 0

END

TO HELP
CS
PRINT ["FLYAWAY" ]
PRINT [ ]
PRINT [BLUE PILOT USES KEYS E,S,
D,X, OR JOYSTICK 1 ]
PRINT [ ]
PRINT [RED PILOT USES KEYS I,J,K
,M, OR JOYSTICK 2 ]
PRINT [ ] PRINT [ ]
PRINT [CONTROLS: ]
PRINT [ ]
PRINT [FASTER: E,I, OR STICKFORW
ARD ]
PRINT [SLOWER: X,M, OR STICKBACK
 ]
PRINT [TURNLEFT: S,J, OR STICKLE
FT ]
PRINT [TURNRIGHT: D,K, OR STICKR
IGHT ]
PRINT [ ] PRINT [ ]
PRINT [SEE WHO CAN TAKE OFF AND
LAND SAFELY FIRST ]
PRINT [ ]
PRINT [BON VOYAGE!! ]

END

TO SETPLANE P
IF :P = 1 THEN MAKE "YS :Y1S ELS
E MAKE "YS :Y2S
TELL :P SC ( 2 * :P + 2 ) SXY :X
S :YS SS 0 SH 90 CARRY 10
END

TO CHECK P
TELL :P
IF SPEED > 10 THEN STOP
IF XCOR = :XS THEN STOP
TEST EITHER YCOR > :UPOFF YCOR <
 :DOWNOFF
IFT CRASH
IFF IF SPEED = 0 THEN WELCOME EL
SE STOP
WAIT 120
SETPLANE WHO
END
```

Continued on p. 74

# The Secret of Personal Record Keeping:

## Implementing DISPLAY AT and ACCEPT AT Without Extended BASIC



**By Paul Karis**

Some of you may have accidentally stumbled upon features of the TI-99/4 that are not described anywhere but which are nonetheless quite helpful. I did . . . and what happily resulted was a way to quickly print text to and accept it from anywhere on the screen *without* having to pass through loops or causing the screen to scroll.

Those of you with Extended BASIC already have this capability with the DISPLAY AT and ACCEPT AT statements. Now, you can have these powerful features in TI BASIC (the language built into the TI-99/4 and 99/4A computers) provided the *Personal Record Keeping* Command Module is inserted. This module which is quite powerful and versatile in itself (see Sept./Oct. issue p. 40-43) will interface with the console's BASIC routines and allow you to use two new statements: CALL D and CALL A. [Those of you without the *PRK* module but who happen to have the *Statistics* module should be able to use that instead—Ed.]

Before getting into the documentation, I should, of course, mention that you can also print anywhere on the screen without CALL D by handling the printing character by character using the subroutine given in the examples in your manuals—i.e., "Character Definition." The drawbacks of that method include lack of speed (the letters appear one by one), more cumbersome programming, and more memory space taken up.

### 1. DISPLAY AT – numerical data

CALL D(R,C,L,V)

| | |
|---|---|
| R = | row number of first character of print line |
| C = | column number of first character of print line |
| L = | maximum length of print line; must be>=1 |
| V = | variable for the value of which is to be printed |

R/C The R(ow) and C(olumn) variables are meaningful with values between 1 and 24, and 1 and 28 respectively (the print field 24x28 is used). Values below the minimum of 1 (0 and negative numbers) are treated as the value 1. Values above the maximum (24 or 28) are automatically subtracted as many times as is required to bring the result between 1 and 24 or 28; this result is then used as the R or C value. This is a nice feature that eliminates many program halts of "BAD VALUE" that often results from careless programming. Data at the end of a screen line is not printed at the beginning of the next screen row as is the case with the CALL HCHAR statement.

L The L position can be used with a fixed number (the maximum meaningful number is 28) or as a variable to which the function can be assigned in numerical data similar to SEG$ in strings.

V Instead of a numerical variable, you can also put a number in this position; it will then be printed on the screen in a position according to the rules above.

*Blauwgras 2*
*3902 AA 'VEENENDAAL*
*The Netherlands*

Example 1

```
100 CALL CLEAR
110 V=326525
120 CALL D(12,10,5,V)
130 GOTO 130
```

Of course you can explain why this program only displays 3265 in the middle of the screen. How would you have to change line 120 to give the full 326525?

### 2. DISPLAY AT – string data

Version 1: CALL D(R,C,L,S$)
Version 2: CALL D(R,C,L,"PAUL W. KARIS")
Version 3: CALL D(R,C,L,CHR$(N))

The variables R,C and L work as described previously under section 1, above.
Here especially, L can be put to good use as a built-in SEG$.

| | |
|---|---|
| Version 1: | the string variables S$ is printed |
| Version 2: | the string between quotes is printed |
| Version 3: | a complicated way of saying CALL HCHAR(R, C, N) that is merely mentioned here as illustration of the possibilities |

Example 2

```
100 CALL CLEAR
110 A$="THIS IS MID-SCREEN"
120 CALL D(12,4,19,A$)
130 GOTO 130
```

Continued on p. 85

WITH MY NEW COMPUTER, NOW I HAVE MORE TIME TO DO WHAT I LIKE BEST—MAKE $$$!
INVENTORY
■FASTER
■UP-TO-DATE
■NO LOST FILES

# Getting Down to Business

By George Struble

This issue, let's consider an inventory system for your computer. I don't have a particular system to review, but I want to discuss what should be involved in an inventory system, and why. This has implications for a number of other applications you might like for your business, such as order processing, accounts payable, and even a general ledger system.

We will address the kind of system you might use in a sales organization—either in a store or through mail or phone order. Some of the description also fits the situation of a raw material inventory or even miscellaneous supplies. Since some of the activities may not fit you if your business is small, be prepared to discount some of the benefits.

First, it is important to know *why* you apply a computer to some task. You should have specific advantages in mind, and know what you have to do to attain those advantages. And, of course, you should be prepared to change your operations as necessary. I have seen organizations that wanted to "put it on the computer" without any clear reason. Often such organizations waste time and resources changing the specifications, design, and operation of a system as they struggle to develop reasons for their system on the fly. Others merely wind up with a system which is a burden to run, with no advantages except an imagined prestige.

On the other hand, I did some work not long ago with a company that was going to get a computer. I expected that payroll would be one of the first applications, as it is in so many companies. But no, they had payroll near the bottom of their list. They were able to do their payroll manually quite well with only 60 or so employees, and had other uses in mind that would give them definite advantages. For them, one of the first priorities was inventory.

What are some of the possible benefits of keeping your inventory records by computer?

1. If you are processing orders by computer, you can improve the efficiency of your warehouse operation in several ways:
   a. The computer can recognize which orders cannot be filled, and thus avoid sending the warehouse crew to look for the items.
   b. The computer can produce "pick slips" or a "picking list" arranged in a sequence to make the picking of the items from the warehouse efficient.
   c. The computer can help manage back-orders; when new stock arrives, the computer automatically scans the file of back-orders and fills any back-orders for the items before allowing new orders a chance.

2. The computer can help you manage your inventory levels effectively and save you money. To do this, you must have good projections of future demand for each item. You can then time your reorders and calculate optimal reorder quantities. At least in theory, you should be able to reduce your working capital tied up in inventory, and at the same time have fewer stock-outs and therefore fill more orders and keep customers happier.

Next, let's consider the information you must keep in your inventory file in order to have an effective inventory system. This file has a record for each product (and perhaps for each size, color, model and style). There is inventory status information: current quantity on hand, quantity on order from vendors, date expected, quantity on back order to customers, and quantity sold since last update. There is also historical demand information such as quantity sold in each month in perhaps the last year. Finally, there is reorder and forecast information—i.e., preferred vendor, vendor's product number, vendor lead time, order quantity, order frequency or reorder point, and demand forecast.

If your computer is processing orders, you also maintain files of back-orders (if permitted). The order processing programs obviously use and update the inventory file. If your computer is not used for processing orders, you must find some other means of updating your inventory data. One of the troubles with this is that your input data to the inventory system are likely to be much less reliable than the order-processing input would be.

An inventory system must include a number of other functions. There are

**"... a company ... went bankrupt, primarily because the order processing and inventory control they did by computer was not accurate.**

**One of the causes ... was the company's lack of understanding of how the system was supposed to work ..."**

**About the Author**

*George Struble, a professor of computer and information science at the University of Oregon, is author of* **Business Information Processing with Basic,** *Addison-Wesley Publishing Co., 1980.*

**TOP QUALITY Professional Programming for your TI-99/4 & TI-99/4A**

*The following are only $5.00 each:*

- Biorhythm
- Music from Numbers
- Draw Shapes
- Sort

*And for $10.00 each we offer:*

- Grade 1 Math
- Word List
- Concentration
- American Baccarat
- Actuary
- Word Processing

*We also have at individual prices:*

| | |
|---|---|
| • Percentage Comparisons | $7.50 |
| • Loan and Pension Calculator | $12.50 |
| • Early Reading | $15.00 |
| • Compu-Strip | $20.00 |
| • Accounting Package (runs on tape — no printer required) | $50.00 |

All prices are U. S. $. Programs airmailed anywhere in the world on tape cassette.

Order from above with money back guarantee or write or call for our *free* catalog with full descriptions.

**ANTHISTLE SYSTEMS & PROGRAMMING LTD.,**
**563 Patricia Drive**
**Oakville, Ontario**
**Canada, L6K 1M4**
*Tel. (416) 845-7959*

simple updates to prices, costs, and warehouse location, as well as addition and deletion of products. There are also inventory adjustments caused by events such as the return of an item from a customer, or the removal of an item for product testing. The function of receiving into inventory is complex: Quantities on hand and on order must be updated. A payable transaction is generated—with its necessary comparison of actual arrival amount with invoiced quantity—so there is an interface with your accounts payable system, if you are using one. Then your system must be sure to trigger the filling of backorders from the new stock *before* letting any new orders have access to it.

Periodically, you must count your physical inventory and adjust your computer inventory accordingly, since you need your inventory file to reflect reality, not wishful thinking. Many events can cause a discrepancy in inventory counts—things such as pilferage, mislabeling, or failure to make the minor adjustments necessitated by the odd-but-authorized removal or replacement of items. The computer should help the physical inventory process by printing the stock list, and by making it easy to adjust the inventory for discrepancies found.

And then there are the functions involved in reordering: About once a week your system should sweep through all products and determine what to reorder. You should of course have the opportunity to over-ride the computer's suggestions, but any such decisions must be recorded (e.g., in quantity on order). Perhaps once a month your system should run some analysis programs that update the demand history and recompute forecasts, reorder quantities etc.

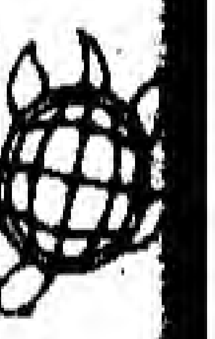
There is even a connection to your general ledger system. After all, inventory is an asset—and any activity that affects the value of that asset should be reflected in your profit and loss, assets and liabilities.

All this is a great deal of work. Not only are there a lot of things to do, but they must be done accurately. I knew a company that went bankrupt, *primarily* because the order processing and inventory control they did by computer was not accurate. They tremendously overstocked some items because the computer said there were none on hand (and of course didn't fill orders because it thought there was no stock), and ran out of stock on other items because the computer thought there was plenty. Naturally, the company couldn't fill those orders either! One of the causes of the snafu was the company's lack of understanding of how the system was supposed to work, how to ensure its accuracy, and how to diagnose inaccuracies.

Continued on p. 84

## Flyaway . . . from p. 71

```
TO WELCOME
PRINT SE [NICE LANDING, ] PILOT
WHO
END

TO STICK S
MAKE "X JOY :S
TELL :S
IF :X < 4 THEN TURNLEFT
IF :X > 6 THEN TURNRIGHT
IF :X = 6 THEN FASTER
IF :X = 4 THEN SLOWER
END

TO MINUS N
IF :N > 10 THEN OUTPUT :N - 1 EL
SE OUTPUT 17
END

TO PLUS N
IF :N < 17 THEN OUTPUT :N + 1 EL
SE OUTPUT 10
END

TO PILOT N
IF :N = 1 THEN OUTPUT :PILOT1 EL
SE OUTPUT :PILOT2
END

TO SLOWER
IF SPEED > 0 THEN SS SPEED - 5
END

TO FASTER
IF SPEED < 100 THEN SS SPEED + 5

END

TO TURNLEFT                TO TURNRIGHT
LT 45                      RT 45
CARRY PLUS SHAPE           CARRY MINUS SHAPE
END                        END

TO SETVAR                  TO LANDED
END                        END

TO CRASH
PRINT SE [NOT SO GOOD, ] PILOT WHO
MAKE "I 10
MAKE "DROP YCOR - ( - 50 )
A:
SY ( - 50 ) + ( :DROP * :I ) / 10
IF :I > 0 THEN MAKE "I :I - 1 GO "A
CARRY 18
SS 0
END
```



## Letters on LOGO . . . from p. 67

I enjoy using the quotation from Albert Einstein that imagination is more powerful than knowledge. No where have I found this to be more evident than in LOGO. You start with your knowledge and your experience. Your imagination, your creativity, and your unique human propensity for illogical and irrational choice, guides you through your discoveries of the unknown. And the marvellous thing about it is that you learn something. However, to better define the relationship of imagination and learning, allow me to cite another quotation.

*"He who has imagination without learning has wings but no feet."*

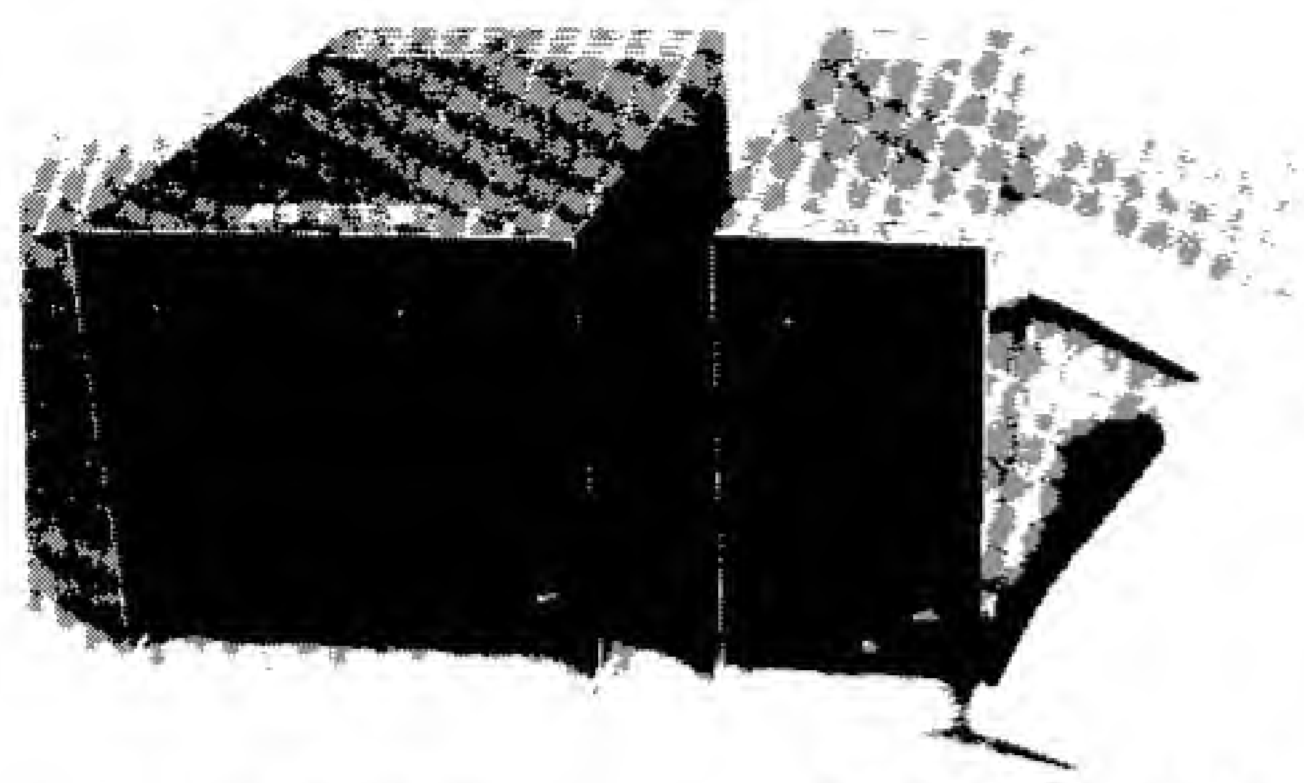
I am very proud to say that our local Texas Turtles have reasonably stable feet on the ground—for young teenagers. But, at the computer, they all have the wings of eagles.

This is the LOGO story that deserves to be told!

James H. Muller
President
Young Peoples' LOGO Association, Inc.

DISK DRIVES

Better Quality
Lower Prices



**DUAL DRIVE**
(Including Cable)
**$599.00**

**SINGLE DRIVE**
(Including Cable)
**$325.00**

These Komputar Works Disk Drives use the TEAC 5¼" Floppy Disk Mechanism and are guaranteed to be compatible with your Texas Instruments TI-99/4 system using the T.I. Disk Controller. They utilize a highly reliable, brushless DC motor to reduce electrical noise. A high precision screw method of head positioning gives greater accuracy. The front doors lock to prevent opening during operation. A rigid, fiberglass frame holds the disk firmly in place, ensuring precise alignment and avoiding problems with heating that can cause expansion and misalignment with conventional disk drives. These drives are backed by a standard 90-day warranty.

KOMPUTAR WORKS HAS THE EXPERTISE YOU NEED FOR YOUR TEXAS INSTRUMENTS TI-99/4A SYSTEMS

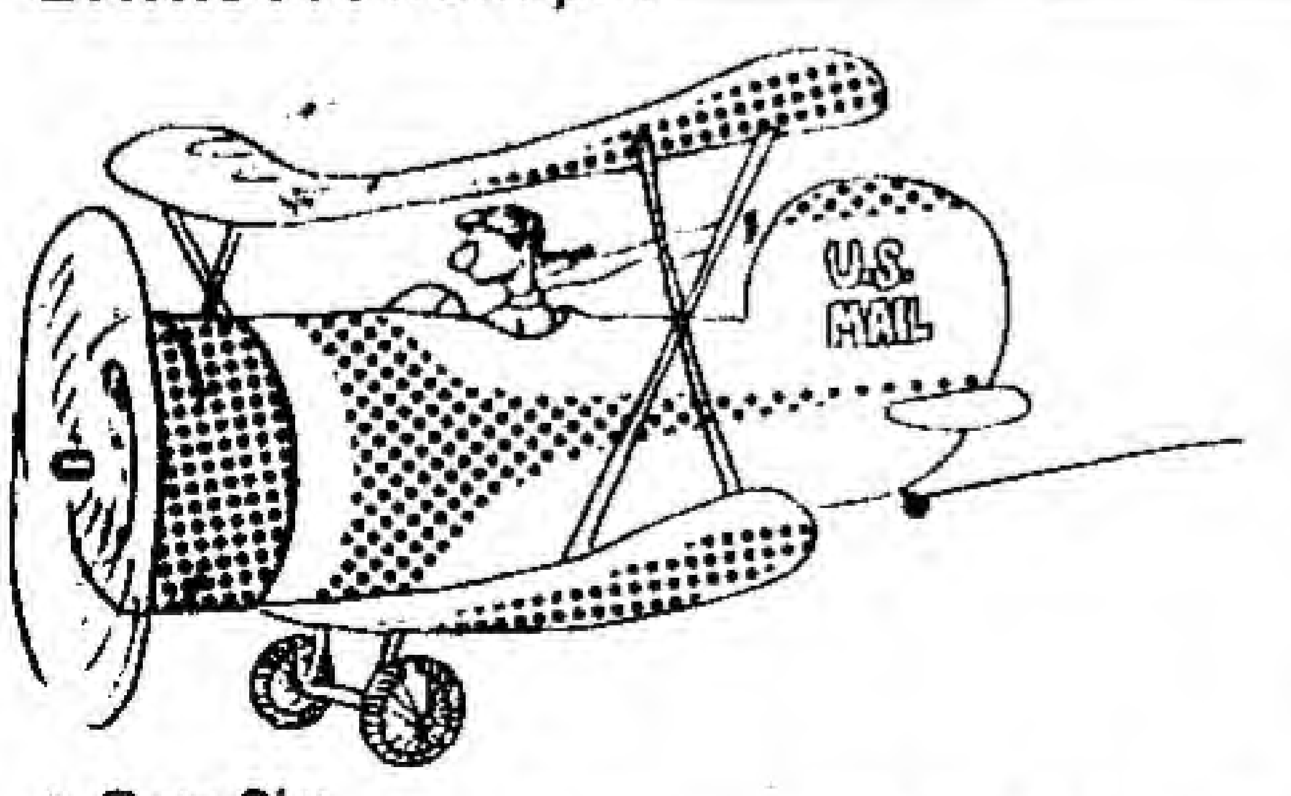
WRITE US

P.O. Box 489
Electric City, WA 99123-0489
509/633-2653

VISA MasterCard

KOMPUTAR WORKS

**Letters . . . from p. 7**



Dear Sir:

A few days after reading about the problem of detecting "sprite" collisions in your third issue (DOGFIGHT, p. 25), I realized that it is similar to a problem encountered in some machine tool controllers where missing an impending collision is "unforgivable."

When dealing with the relative motions of two or more objects, it is necessary that they all have a common variable. In "real time" applications, the use of the time parameter itself is often overlooked because it is not explicitly stated in specifications. Sprite graphics is such a "real time" application.

Now, when two sprites are in straight line motion, there is a unique point in time when the distance between them will be at a minimum. Obviously, if a collision is to occur, it will occur at this time.

Therefore the possibility of a collision between two sprites need not be checked for continuously, but only once, for each CALL SPRITE statement.

But, the technique relies on the discipline of Analytic Geometry and therefore must be presented to programmers as a "plug-in" formula.

Although, in some cases, this "look ahead" technique might improve the performance of Extended Basic applications, its real advantage is manifested in Assembly Language where 16 bit integer arithmetic is sufficient for the variables. Unless I terribly misunderstand something, the arithmetic for the time-collision projection would take around 250 microseconds.

Eldon Kerfoot
Hacienda Hts., CA

*Your point is well taken, Eldon. However, your suggestion that the coincidence test be performed only once for each CALL SPRITE statement is not true if sprites are allowed to wrap around the display; in those cases, a new TNEAR calculation is required whenever one of the sprites enters at the edge of the screen. Also, you'd need to recalculate TNEAR any time a sprite's direction or velocity changed (via CALL MOTION). The problem with your approach as a whole is the difficulty of implementing it. Suppose the programmer coded the TNEAR and accompanying DIST calculations immediately following the creation of the sprites. What should the program do if DIST at time TNEAR is found to be within the collision range? Of course, we'd like the program to show an explosion at time TNEAR; but how is this accomplished? The program could just wait around until time TNEAR arrives, which would mean that the programmer would only have to know the execution times of a couple of statements (like FOR and NEXT in XBASIC or DEC and JNE in AL, for example); this wouldn't be too hard to grasp. But what if the program has other things to do in the meantime? The programmer would have to be aware of the execution times of any statements used so that he could know when TNEAR has arrived.*

*As you imply, this is not practical in Extended BASIC, not only because the programmer would have to know the execution times of the instructions, but also because many statements take so long to execute that TNEAR might be long past before the program could show an explosion. Adding timing logic is certainly possible, although somewhat difficult, in Assembly Language.*

Dear Sir:

Congratulations! You have managed to produce the *best* computer magazine I have seen (and I presently read Popular Computing, Personal Computing, Microcomputing and Creative Computing). Your articles are super, both content and style. Keep up the good work. I have already found two new subscribers for you.

John Puisar
Norfolk, VA

Dear Sir:

I'd like to give you a solution to the automatic load program for Extended *BASIC*, which I've had working for some time. First you must understand that the RUN statement uses a character constant as its argument. There's no way around that. To make it variable you either have to change the value of the constant or write a new program. It's impossible to write PROGRAM-type files in BASIC so we'll have to do for now with rewriting the constant. This requires use of the Memory Expansion. The following code solves the problem. The lines before 600 should define the program-name.

```
100 PRG$="DSK1.RUN THIS"
600 CALL INIT
610 FOR Y=1 TO 15
620 S$=SEG$(PRG$&CHR$(0)&"XXXXX
    XXXX",Y,1)
```

Continued on p. 91

Super Language . . . from p. 60

## Listing 3.

```
 99/4 ASSEMBLER
VERSION 1.1                                                  PAGE 0001
 0001                    IDT   'SHOOTA'
 0002                    DEF   SHOOTA
 0003                    REF   VMBW,VSBW,VWTR,VMBR
 0004 0000         WS    BSS   32
 0005 0020    7C   SAL   BYTE >7C,>01,>80,>06    SPRITE 1 LOCN AND COLOR
      0021    01
      0022    80
      0023    06
 0006 0024    01         BYTE >01,>7C,>81,>01    SPRITE 2 LOCN AND COLOR
      0025    7C
      0026    81
      0027    01
 0007 0028    0D         BYTE >0D                TERMINATOR
 0008 0029    FF   SHAPE BYTE >FF,>81,>BD,>A5,>A5,>BD,>81,>FF TARGET
      002A    81
      002B    BD
      002C    A5
      002D    A5
      002E    BD
      002F    81
      0030    FF
 0009 0031    18         BYTE >18,>18,>18,>18,>18,>18,>3C,>18 ARROW
      0032    18
      0033    18
      0034    18
      0035    18
      0036    18
      0037    3C
      0038    18
 0010 0039    81   HITSHP BYTE >81,>42,>24,>18,>18,>24,>42,>81 HIT SHAPE
      003A    42
      003B    24
      003C    18
      003D    18
      003E    24
      003F    42
      0040    81
 0011 0041    00   SPEED  BYTE >00,>64,>00,>00   SPRITE 1 VELOCITY
      0042    64
      0043    00
      0044    00
 0012 0045    7F          BYTE >7F,>00,>00,>00   SPRITE 2 VELOCITY
      0046    00
      0047    00
      0048    00
 0013 0049    00   H00    BYTE >00
 0014 004A    02   H02    BYTE >02
 0015 004B         Y1     BSS  1
 0016 004C         X1     BSS  1
 0017 004D         DUMMY  BSS  2
 0018 004F         Y2     BSS  1
 0019 0050         X2     BSS  1
 0020 0051    03   H03    BYTE >03
 0021 0052    07   H07    BYTE >07
 0022                     EVEN
 0023 0054 0020    H0020  DATA >0020
 0024 0056 02E0    SHOOTA LWPI WS
      0058 0000'
 0025              * FILL SCREEN WITH BLANKS.
```

```
 99/4 ASSEMBLER
VERSION 1.1                                                  PAGE 0002
 0026 005A 04C0           CLR  0              VDP RAM SCREEN HOME
 0027 005C 0201           LI   1,>2000        BLANK IN MSB OF R1
      005E 2000
 0028 0060 0420    BLNKIT BLWP @VSBW          WRITE BLANK
      0062 0000
 0029 0064 0580           INC  0
 0030 0066 0280           CI   0,768          DONE?
      0068 0300
 0031 006A 11FA           JLT  BLNKIT         NOT YET
 0032              * SET UP VDP REGISTER 1.
 0033 006C 0200           LI   0,>01E0        NORMAL SIZED SPRITES
      006E 01E0
 0034 0070 0420           BLWP @VWTR
      0072 0000
 0035              * SET UP SPRITE ATTRIBUTE BLOCK.
 0036 0074 0201    DEFSPR LI   1,SAL          R1->MY ATTRIBUTE LIST
      0076 0020'
 0037 0078 0200           LI   0,>300         R0->ADDRESS OF VDP SAB
      007A 0300
 0038 007C 0202           LI   2,9            9 BYTES TO WRITE
      007E 0009
 0039 0080 0420           BLWP @VMBW          WRITE TO VDP RAM
      0082 0000
 0040              * LOAD SPRITE DEFINITIONS.
 0041 0084 0201           LI   1,SHAPE        R1->MY SPRITE SHAPES
      0086 0029'
 0042 0088 0200           LI   0,>400         ADDRESS OF FIRST SPRITE
      008A 0400
 0043 008C 0202           LI   2,16           16 BYTES TO MOVE
      008E 0010
 0044 0090 0420           BLWP @VMBW          WRITE TO VDP RAM
      0092 0082'
 0045              * SET UP SPRITE MOTION TABLE.
 0046 0094 0200           LI   0,>780         R0->MOTION TABLE IN VDP RAM
      0096 0780
 0047 0098 0201           LI   1,SPEED        R1->MY SPEED DATA
      009A 0041'
 0048 009C 0202           LI   2,8            8 BYTES TO MOVE
      009E 0008
 0049 00A0 0420           BLWP @VMBW          WRITE
      00A2 0092'
 0050              * SET NUMBER OF MOVING SPRITES.
 0051 00A4 D820           MOVB @H02,@>837A  2 MOVING SPRITES.
      00A6 004A'
      00A8 837A
 0052              * MAKE SPRITES MOVE BY INTERRUPT FROM 9901 I/O BOARD.
 0053 00AA 0300    MOVEIT LIMI 2              ENABLE VDP INTERRUPT
      00AC 0002
 0054              * CHECK FOR COINCIDENCE.
 0055 00AE 0300           LIMI 0              DISABLE VDP INTERRUPT
      00B0 0000
 0056              * GET SPRITE POSITIONS.
 0057 00B2 0200           LI   0,>300         R0->Y OF SPRITE 1 IN VDP RAM
      00B4 0300
 0058 00B6 0201           LI   1,Y1           BUFFER FOR READ
      00B8 004B'
 0059 00BA 0202           LI   2,6            6 BYTES TO READ
      00BC 0006
 0060 00BE 0420           BLWP @VMBR          READ FROM VDP RAM
```

```
 99/4 ASSEMBLER
VERSION 1.1                                                  PAGE 0003
      00C0 0000
 0061              * CHECK COLUMNS FOR X1<=X2+3<=X1+7.
 0062 00C2 B820           AB   @H03,@X2       X2=X2+3
      00C4 0051'
      00C6 0050'
 0063 00C8 7820           SB   @X1,@X2        X2=X2-X1
      00CA 004C'
      00CC 0050'
 0064 00CE [illegible]    JLT  MOVEIT         NO HIT IF RESULT < 0
 0065 00D0 [illegible]    CB   @X2,@H07       COMAPRE TO 7
      00D2 0050'
      00D4 0052'
 0066 00D6 15E9           JGT  MOVEIT         NO HIT IF RESULT > 7
 0067              * CHECKS ROWS FOR Y1<=Y2+7<=Y1+7.
 0068 00D8 B820           AB   @H07,@Y2       Y2=Y2+7
      00DA 0052'
      00DC 004F'
 0069 00DE 7820           SB   @Y1,@Y2        Y2=Y2-Y1
      00E0 004B'
      00E2 004F'
 0070 00E4 11E2           JLT  MOVEIT         NO HIT IF RESULT < 0
 0071 00E6 9820           CB   @Y2,@H07
      00E8 004F'
      00EA 0052'
 0072 00EC 15DE           JGT  MOVEIT         NO HIT IF RESULT > 7
 0073              * HIT.
 0074              * CHANGE SPRITE DEFINITIONS.
 0075 00EE 0201           LI   1,HITSHP       R1-> HIT SHAPE
      00F0 0039'
 0076 00F2 0200           LI   0,>400         R0->VDP RAM
      00F4 0400
 0077 00F6 0202           LI   2,8            8 BYTES TO LOAD
      00F8 0008
 0078 00FA 0420           BLWP @VMBW          WRITE TO VDP RAM
      00FC 00A2'
 0079              * WAIT TO LET BLOW UP BE SEEN.
 0080 00FE 0203           LI   3,10           OUTER LOOP CTR
      0100 000A
 0081 0102 0202    LOOP2A LI   2,12000        LOOP COUNTER
      0104 2EE0
 0082 0106 0602    LOOP2  DEC  2              DECREMENT
 0083 0108 16FE           JNE  LOOP2          WAIT MORE
 0084 010A 0603           DEC  3              DECREMENT OUTER CTR
 0085 010C 16FA           JNE  LOOP2A         WAIT MORE
 0086 010E 10B2           JMP  DEFSPR         SATRT OVER
 0087                     END  SHOOTA
```

```
 99/4 ASSEMBLER
VERSION 1.1                                                  PAGE 0004
   ' BLNKIT   0060    ' DEFSPR  0074    ' DUMMY   004D    ' H00     0049
   ' H0020    0054    ' H02     004A    ' H03     0051    ' H07     0052
   ' HITSHP   0039    ' LOOP2   0106    ' LOOP2A  0102    ' MOVEIT  00AA
   ' SAL      0020    ' SHAPE   0029    D SHOOTA  0056    ' SPEED   0041
   E VMBR     00C0    E VMBW    00FC    E VSBW    0062    E VWTR    0072
   ' WS       0000    ' X1      004C    ' X2      0050    ' Y1      004B
   ' Y2       004F
   0000 ERRORS
```

## Listing 4.

```
100 CALL CLEAR
110 REM DEFINE SPRITES
120 CALL CHAR(142,"FF81BDA5A5BD81FF")
130 CALL CHAR(143,"181818181818 3C18")
140 CALL CHAR(141,"8142241818244281")
150 CALL SPRITE(#1,142,7,124,1,0,100)
160 CALL SPRITE(#2,143,2,1,124,127,0)
170 REM TEST FOR HIT.
180 CALL COINC(#1,#2,10,HIT)
190 IF HIT=0 THEN 180
200 CALL MOTION(#1,0,0)
205 CALL MOTION(#2,0,0)
210 CALL PATTERN(#1,141)
221 FOR DELAY=1 TO 50
222 NEXT DELAY
230 GO TO 150
240 END
```

## Understanding An Assembler Listing

The Assembly Language listing which follows was output by the 99/4A Assembler. You'll notice that the Assembler has added a page number and short title at the top of each page, and added a cross-reference list and number-of-errors-found-during-assembly message to the end. The cross-reference list shows the locations of the symbols used in the program relative to the beginning of the program. The line numbers in the first column of the listing were supplied by the Editor when the program was input and passed along by the Assembler. The second column of the listing shows the relative memory location where each statement or data area will reside during program execution. The third column was also supplied by the Assembler, and shows the machine language generated by the Assembly Language statement to the right. The machine language is expressed in hexadecimal notation with one word per line. The Assembly Language source program itself starts in the fourth column, which contains the labels. The fifth column contains the source program opcodes, and the sixth column contains the operands. The seventh column contains comments, and other comments are sprinkled throughout the program with asterisks in column 1. *Only the fourth through seventh columns comprise the Assembly Language source program; this is the only part entered by the programmer.* The assembler generates the rest.

Continued on p. 85

*Custom Peripherals from* **COMPUTECH**

**–We Mak'em to Order–**
Separate your Peripherals with custom, shielded, I/O cables from *COMPUTECH.*

```
100  Y$ = Your Choice
110  Length = 1 Ft.*Y$
120  Price = 24.00+(Length*4.00)
```

**417-869-1684**
Dealer Inquiries Invited

**Numeric Data Entry is a Breeze With a Computech "SPEED KEY"**

Designed for the TI 99/4 or /4A.

The "SPEED KEY" docks directly into the console (RS232 NOT Required). Required support software is resident in the "SPEED KEY" and no external power supply is required. Comes complete and ready to use. Price 169.95 + Shipping. (Allow 3-4 wk. Del.)

**COMPUTECH** 209 E. Walnut • Springfield, MO 65806

---

**CUMBERLAND TECHNOLOGY**
10 Wagner Drive
Carlisle, PA 17013

**99/4(A) Programs**

- ENGINEERING
- MATH
- PROGRAMMING AIDS
- GAMES
- Many programs written in Assembly Language

*Please send name and address for a current list*

---

**Need Software?**

- Educational
- Entertainment
- Business
- Mathematics

Send $1.00 (refundable on order) for our catalog

**DATA SYSTEMS**
2214 W. IOWA
Chicago, IL 60622

---

software
NORTON
software

SOFTWARE FOR THE TI 99/4 COMPUTER
Featuring
Two Arcade Speed, Extended Basic Games

**ATTACK-MAN**
This game is similar to the pursuit games that you find in the arcade except that this one is more of a challenge. Collect points while running through a maze being pursued by a gang of lovable hungry eaters. Can you make it to the energy pellet (most likely not) so you can turn on them! YOU CAN FIND OUT FOR .......... 12.95

**CROSS COUNTRY CAR RALLY**
Inspired by the cannonball run, race across the country, avoiding police, tourists, your rivals and a few surprizes. This high resolution graphic road adventure can be yours for less than a speeding ticket. FINE .......... 12.95

*THAT'S LESS THAN 52 QUARTERS!!!!!!*

**Send to — Norton Software**
**Box 575 Picton, Ontario K0K 2T0**

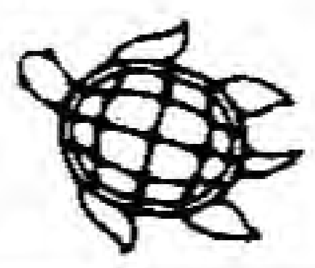
---

**Extending LOGO . . . from p. 69**

```
CALL READLINE "ADJ
PRINT [TYPE A LIST OF VERBS AND
THEN PRESS ENTER. NOW WATCH. ]
CALL READLINE "VERBS
GRAMMAR :ART :NOUNS :ADJ :VERBS
END

TO GRAMMAR :ART :NOUNS :ADJ :VER
BS
TYPE SELECT ( NUMB ( COUNT :ART
) ) :ART
SPACE
TYPE SELECT ( NUMB ( COUNT :NOUN
S ) ) :NOUNS
SPACE
TYPE SELECT ( NUMB ( COUNT :VERB
S ) ) :VERBS
SPACE
TYPE SELECT ( NUMB ( COUNT :ADJ
) ) :ADJ
SPACE
TYPE SELECT ( NUMB ( COUNT :NOUN
S ) ) :NOUNS
PRINT ".
WAIT 30
GRAMMAR :ART :NOUNS :ADJ :VERBS
END
```

SENTENCES can be made a better grammarian by adding distinctions of number and gender where appropriate; it can be made a more sophisticated language generator if GRAMMAR is altered to allow for conjunctions and subordinate clauses. All of these changes and more can be programmed by students as they learn both the specifics of grammar[2] and mathetics of LOGO.

[2] Papert would probably argue that most students know the grammar which schools attempt to teach, but that the students do not have verbal labels for syntactical rules and parts of speech (and do not see the relevance of the labels) once they are told them. A sentence generator program can make grammar "speech syntonic."



---

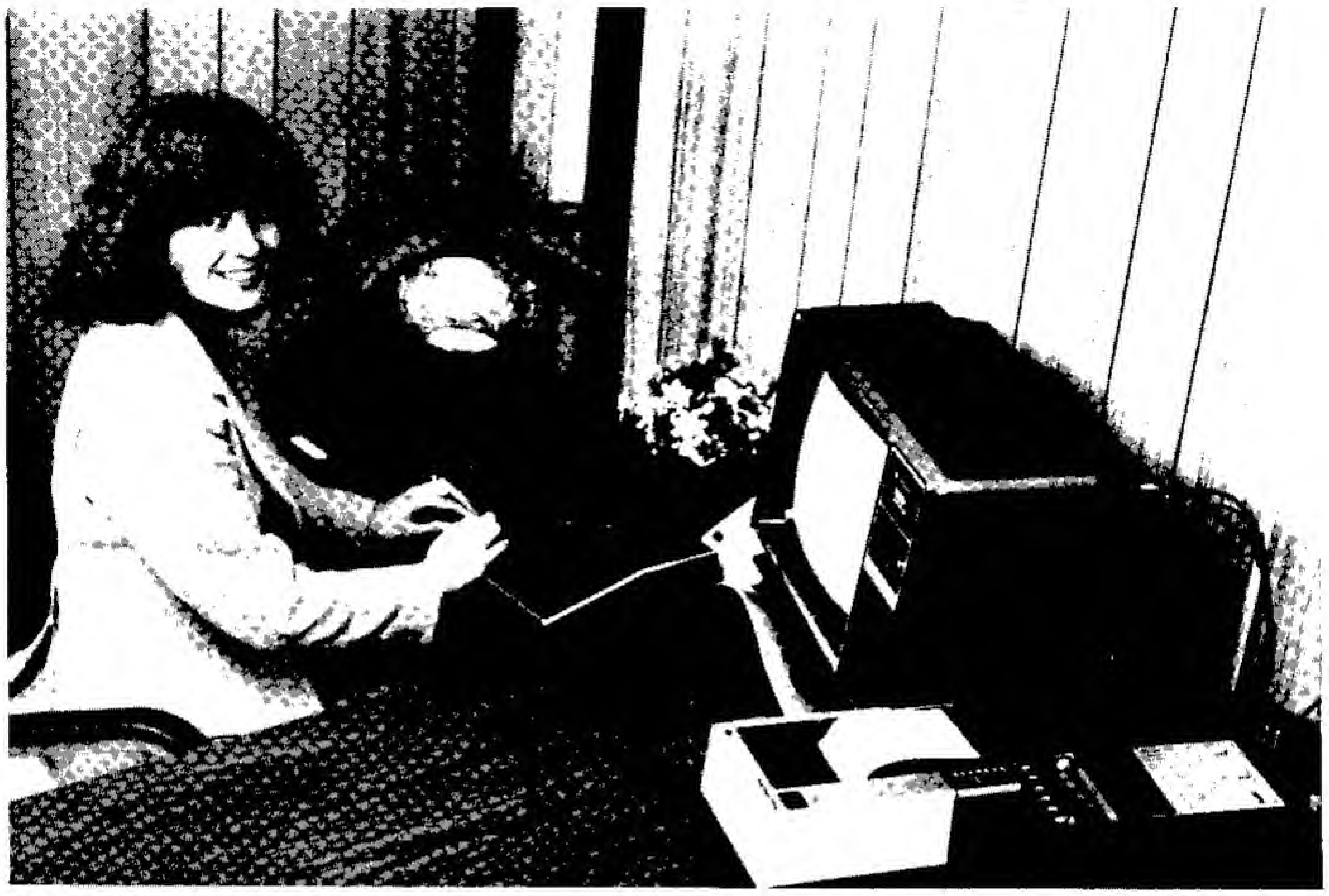
**Computer Chess Corner**
will resume next issue.

**SOLUTIONS TO THE PROBLEMS IN THE LAST ISSUE:**

*Problem No. 1:*
1. F5 - F6 check!   H5 - G4
2. F7 - E6 Check   D7 - E8
3. F6 - F7 checkmate

*Problem No. 2*
1. . . . E4 - G3 check!
2. H2 - G3   H4 - G3 check.
3. H1 - G1   G4 - F7 (checkmate at H1 is threatened)
4. F1 - F2 (Forced)   H8 - H1 check !!
5. G1 - H1   G3 - F2 and wins since white cannot prevent black from promoting his pawn at F2 to a queen. A queen against a rook is a crushing material advantage.

**Now, you can separate your TI 99/4 Computer from your TI Peripherals**



BETTER CONVENIENCE • EASE OF OPERATION • VERSATILITY

Our (2') Two Foot Cable is made to interconnect your Texas Instruments TI 99/4 Computer with almost any combination of Printer, RS 232, Tape, or Controller.

---

**ORDER FORM**

*SEND ORDER WITH PAYMENT TO:*
**RCL COMPUTERS**
411 Allegheny River Blvd.
Oakmont, PA. 15139
**(412) 828-4301**

**COMPUTERS**

SHIP THE FOLLOWING:

__________ Unshielded 2' Cable At $31.95 ea.

__________ Unshielded 3' Cable At $36.95 ea.

Shielded Cable Prices Upon Request

☐ Check No. ________ Amount ________

☐ Money Order ________ Amount ________

*Price F.O.B. Pittsburgh, PA.*
*PA. Deliveries add 6% Sales Tax*
*Allow 3-4 weeks delivery*

SHIP TO:

NAME ________________

ADDRESS ________________

CITY ________ STATE ____ ZIP ____

Fun & Games . . . from p. 29

```
260 DX=1
270 DY=0
280 GOTO 320
290 IF K<>83 THEN 150
300 DX=0
310 DY=-1
320 GOTO 360
330 CALL JOYST(1,JA,JB)
340 DX=-JB/4
350 DY=JA/4
360 CALL HCHAR(X,Y,32)
370 X=X+DX
380 Y=Y+DY
390 X=INT(24*((X-1)/24-INT
      ((X-1)/24)))+1
400 Y=INT(32*((Y-1)/32-INT
      ((Y-1)/32)))+1
410 CALL HCHAR(X,Y,42)
420 GOTO 150
430 STOP
```

## Detecting a Crash

Probably the most common way of determining if your moving object hit some obstacle in position X,Y is by using CALL GCHAR(X,Y,C). The C value returned is the character number occupying position X, Y on the screen. For example, you may then test if C=32 (space); if so the program could continue. But if C=96 (one type of object), the program would branch one way, and if C=99 (another object) the program would branch another way—all with appropriate sounds and graphics.

Another method of determining the character in a certain position is to have the screen positions in an array and each array element would contain information about the character in that position. For example, you may have an array A(24, 32) for the 24 rows and 32 columns of the screen. Each element of A could be zero for a space and 1 for a block in a maze. Your testing statement would look like:

```
200  IF A(X,Y)=1 THEN 240
```

Which means if the position X ,Y is a block then branch to Line 240 where a crashing noise is made and appropriate action takes place. Note that by using OPTION BASE 1, you will eliminate Row 0 and Column 0 and save memory space.

## Extended BASIC

TI Extended BASIC enables you to create faster moving arcade-type games that make this Command Module worth the money. Multiple-statement lines improve the execution speed. 28 sprites can move around the screen in a myriad of directions to simulate anything from wild animals to outer space. If you have Extended BASIC, take advantage of the moving sprites!

Another big plus for Extended BASIC is the DISPLAY AT and ACCEPT AT capability. DISPLAY AT(row, col) SIZE(S) : G allows printing of titles quickly in your game without messing up graphics or scrolling. You may print the amount of fuel left, time, the score, or messages. ACCEPT AT( X , Y ) VALIDATE (DIGIT) BEEP SIZE(S) : Z helps eliminate user input error where CALL KEY is not appropriate (user needs to press more than one key).

All games use some IF-THEN-ELSE logic, and Entended BASIC allows some intricate possibilities combining statements and line numbers

I'm going to refer you to past issues of *99'er Magazine* for some good programming examples using Extended BASIC in games. Every Extended BASIC owner should have keyed in "Sprite Chase" from the July/August 1981 issue. This fun game shows how to MAGNIFY and define randomly moving sprites. Lines 460-530 show how to program the arrow keys (including diagonal movement). A time clock is also printed to keep the game challenging.

"Dogfight" in the September/October 1981 issue is another captivating game, and you should have keyed it in by now and had hours of fun with it. Now get the magazine back out and take a look at the listing and read the accompanying article. An explanation is given of all the Extended BASIC features used in the game and specifically what they do.

Detecting crashes of moving sprites is done with CALL COINC — finding the coincidence of two objects or of any objects within a certain tolerance. You will need to experiment quite a bit with your game program to make sure it works the way you want it to. Coincidence is reported only at the exact moment the CALL COING statement is executed, so it needs to be placed at strategic places in the program. The coincidence depends on the tolerance allowed and also the velocity of the moving objects. Remember to minimize the number of statements and the logic between CALL COINC statements.

## Do It!

I've presented some fundamental hints and ideas for programming; now it's your turn to put on your thinking cap, turn on the computer, and have fun writing your *own* games!

## DEBUGS ON DISPLAY

99'er Program Bug

**The following is a list of changes to programs that have already appeared in our first three issues. Some are due to the slightly less available memory in the new TI-99/4A; others are simply efficiency enhancements; and the rest is our attempt to erradicate that ubiquitous 99'er Program Bug!**

### Volume 1, No. 1

**p. 69 Music File Player**

```
(Change) 240 DIM P(530, 2)
(Change) 290 FOR I=0 TO 530
```

**p. 73 Tiny Math 1**

Lines 940-950 ask for the time you want, from 1 to 7 seconds. Lines 1590-1640 do not give you that number of seconds, but much less time. For example, for T=7, the time delay is about 2.7 seconds. Use CALL SOUND instead of a FOR-NEXT counting loop.

```
(Change) 1610 FOR LOOP=1
              TO T
(Add)    1612 CALL SOUND
              (1000,9999,30)
(Same)   1620 NEXT LOOP
(Add)    1622 CALL SOUND
              (1,9999,30)
```

Lines 1240-1250 ask if you want to play again. If you answer "Y" for yes, then variables are initialized and you return to Line 440. The screen that prints what the five shapes are does not show the shapes because they have been made transparent and not returned to visible. Also, Character 96 will be whatever symbol was used in the previous game.

```
(Add)    2995 CALL COLOR
              (9,2,1)
(Add)    2996 CALL CHAR
              (96,G$(1))
```

**p. 89 Homework Helper: Fractions**

All program numbers in the "Program Explanations" and the accompanying article need to be increased by 30 to correctly correspond to the listing.

After each problem is completed, the student is instructed to
PRESS 1 FOR NEXT PROBLEM
PRESS 2 TO STOP

The program works fine if the student does press "1" or "2"; however, if any other key is pressed, the screen prematurely clears and the program still waits for "1" or "2". The following changes need to be made.

```
(Change) 1310 IF K<>49
              THEN 1330
(Change) 1320 CALL CLEAR
(Add)    1325 GOTO 780
(Change) 1620 IF K<>49
              THEN 1640
(Change) 1630 CALL CLEAR
(Add)    1635 GOTO 1350
(Change) 1880 IF K<>49
              THEN 1900
(Change) 1890 CALL CLEAR
(Add)    1895 GOTO 1660
(Change) 2620 IF K<>49
              THEN 2640
(Change) 2630 CALL CLEAR
(Add)    2635 GOTO 2400
(Change) 2900 IF K<>49
              THEN 2920
(Change) 2910 CALL CLEAR
(Add)    2915 GOTO 2660
(Change) 3230 IF K<>49
              THEN 3250
(Change) 3240 CALL CLEAR
(Add)    3245 GOTO 2940
(Change) 5470 IF K<>49
              THEN 5490
(Change) 5480 CALL CLEAR
(Add)    5485 GOTO 5070
```

The program works fine as is and gives correct answers but does not use the efficiency technique for simplifying as described in the article.

```
(Change) 1460 IF DS/2<>INT
              (DS/2) THEN 1480
(Change) 1470 IF NS/2=INT
              (NS/2) THEN 1490
```

### Volume 1, No. 2

**p. 25, Battle At Sea**

```
(delete) 180 RANDOMIZE
(Add)    2385 RANDOMIZE
(Change)
         5050 IF SCORE=5
         THEN 5100
(Change) 210 PRINT TAB(7);
         "BATTLE AT SEA"
```

**p. 62, Civil Engineering**

Delete REMark statements 110, 120, 140, and 160

### Volume 1, No. 3

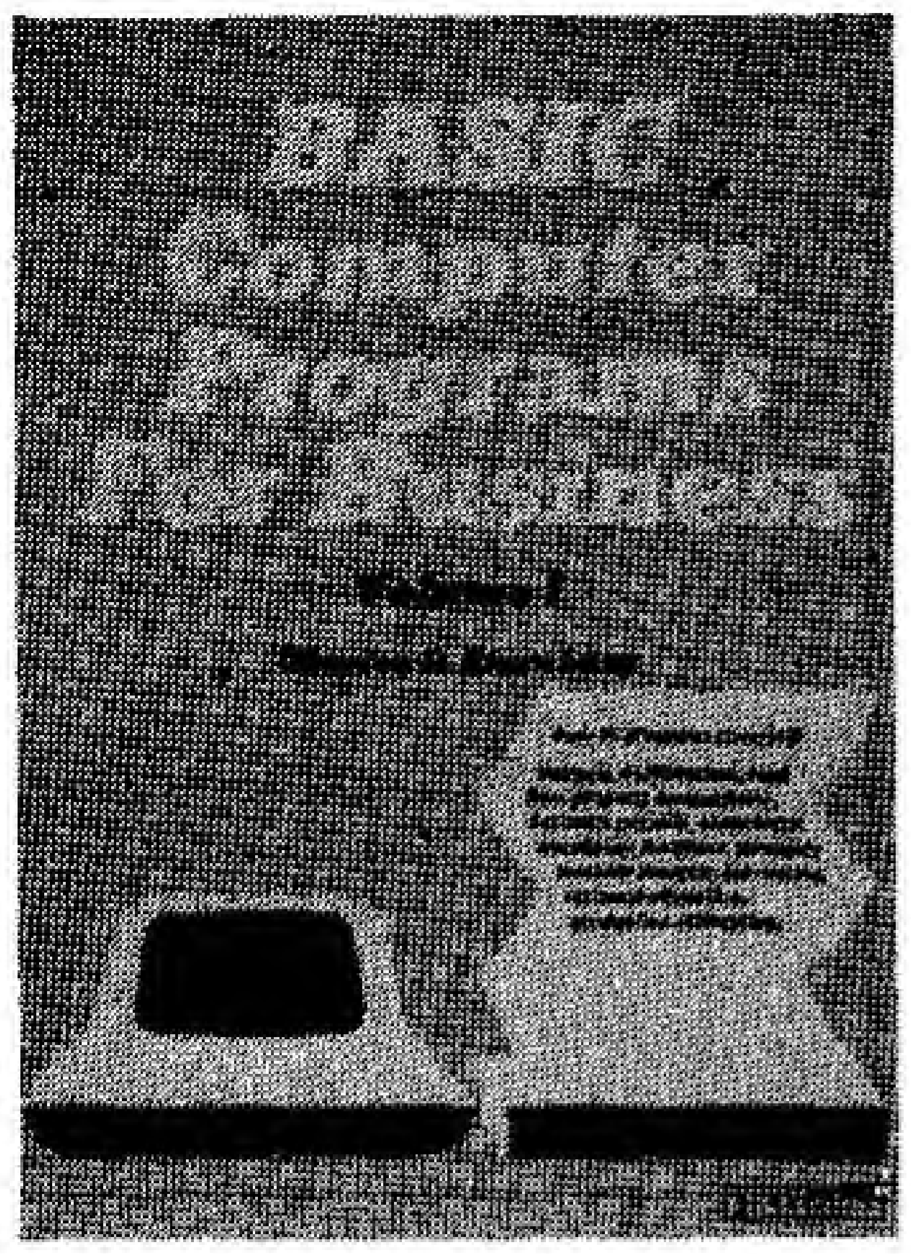
**p. 27, Space War**

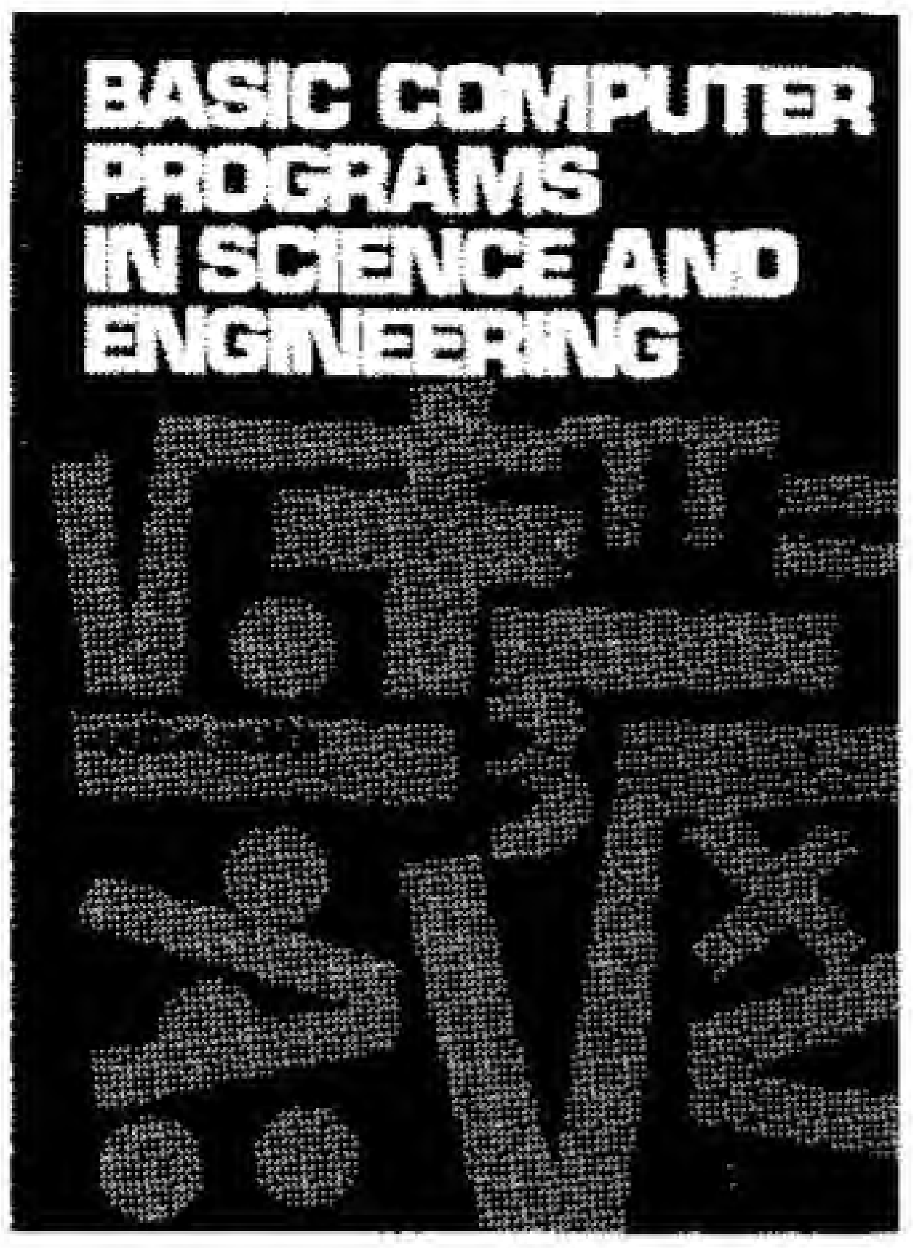
```
(Change) 2700 GOTO 2160
```

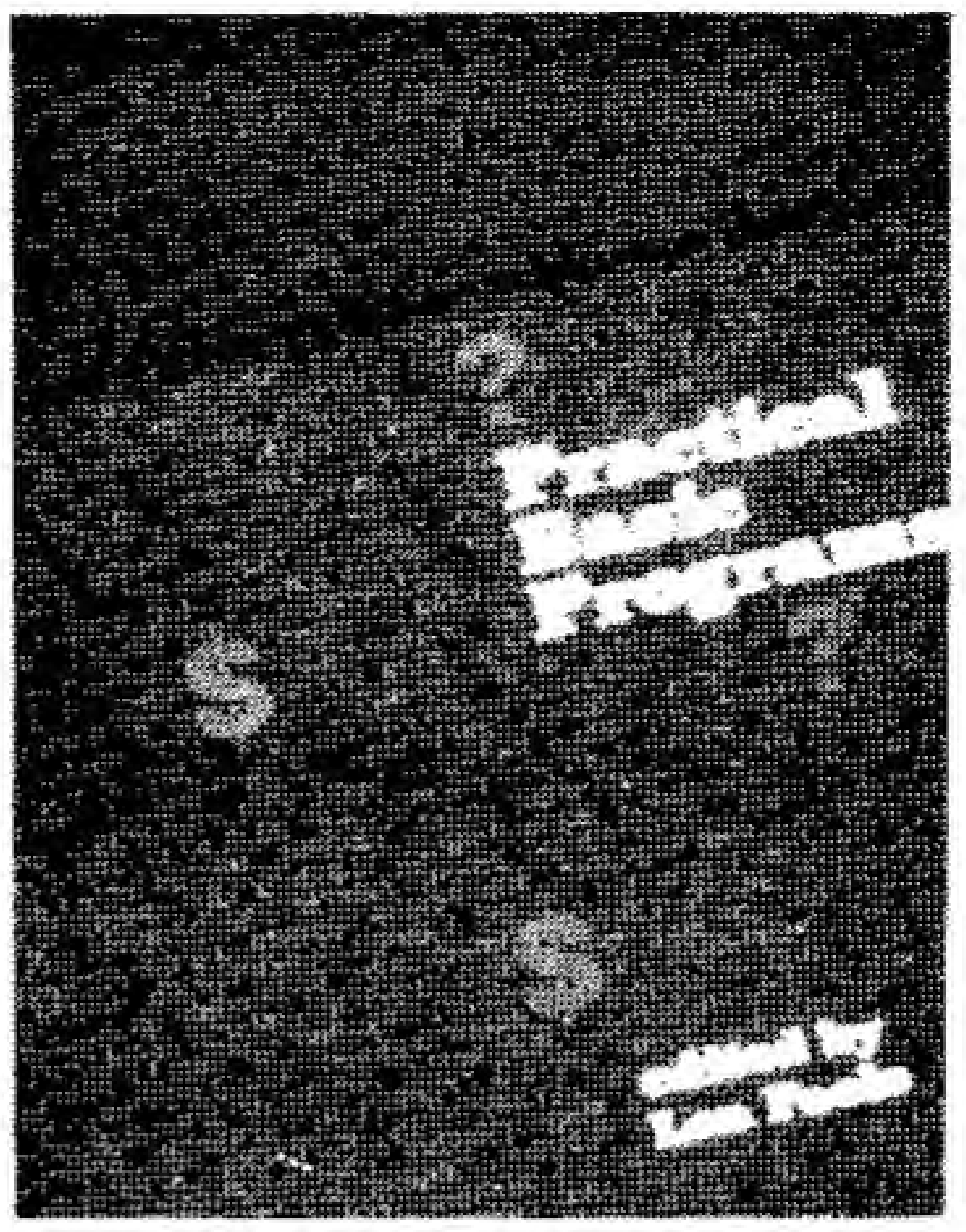
**p. 34, Checkbook Balance**

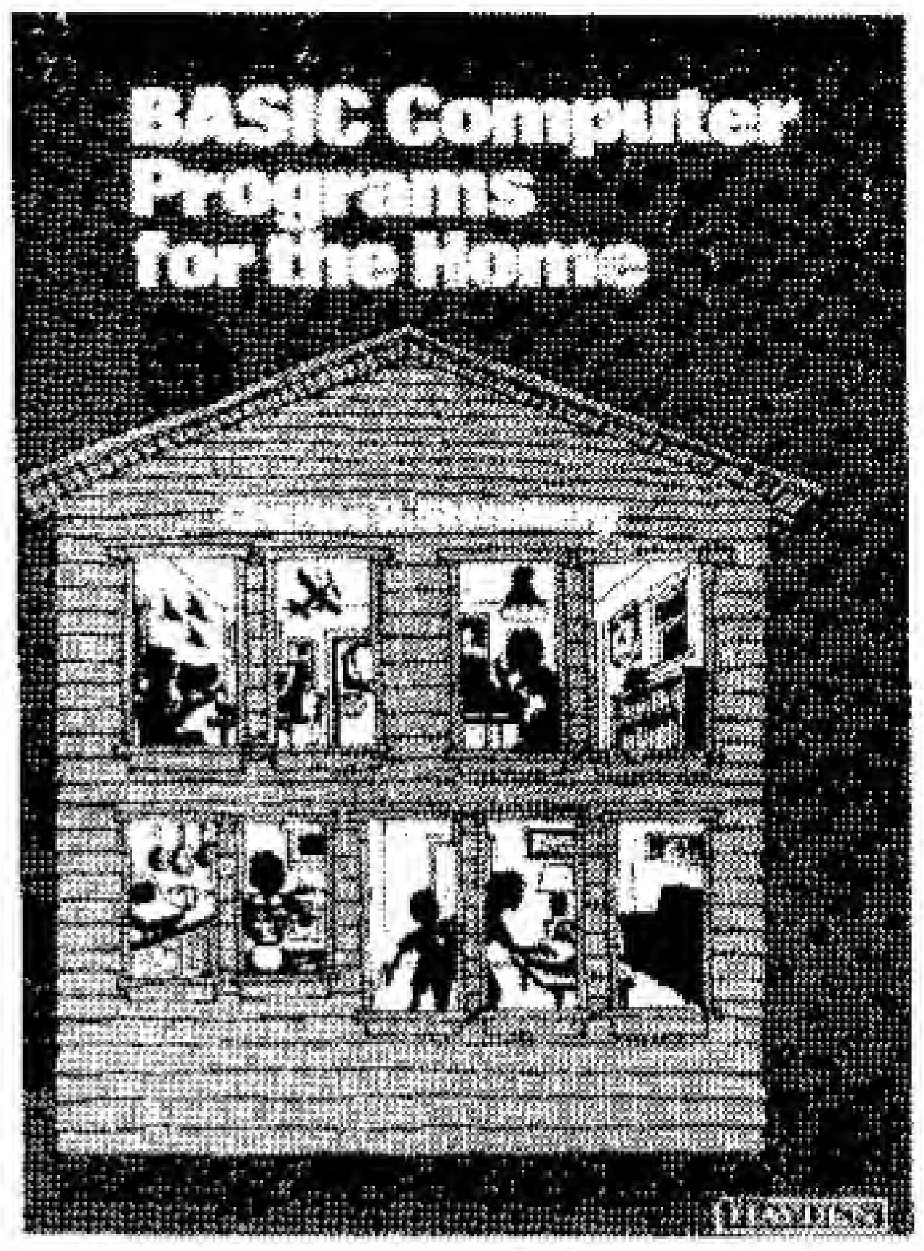
```
(Change)
230 CTOTAL=CTOTAL+
    CAMT
```

# 99'er BOOKSTORE









## BASIC COMPUTER PROGRAMS FOR BUSINESS : VOL. 1

By Charles D. Sternberg.
A must for small businesses utilizing micros as well as for entrepreneurs, these two volumes provide a wealth of practical business applications. Each program is documented with a description of its functions and operation, a listing in BASIC, a symbol table, sample data, and one or more samples.

Volume 1 contains over 35 programs covering: budgets, depreciation, cash flow, property comparisons, accounts payable, order entry, warehouse locations, inventory turnover analysis, job routing, resource allocation, production scheduling, etc.

*volume 1, paper,* **$11.95**
*1980, 384 pages, 7 x 10*

## BASIC COMPUTER PROGRAMS IN SCIENCE AND ENGINEERING

By Jules H. Gilder.
Save time and money with this collection of 114 ready-to-run BASIC programs for the hobbyist and engineer. There are programs to do such statistical operations as means, standard deviation averages, curve-fitting, and interpolation. There are programs that design antennas, filters, attenuators, matching networks, plotting, and histogram programs. There is even a justified typing program that can be used in typesetting. All programs in the book have been tested and are fairly universal; so you should have no difficulty running them on your system. You won't find anywhere a more comprehensive collection of usable, ready-to-run BASIC programs!

*paper,* **$10.95**
*1980, 160 pages, 6 x 9, illus.*

## PRACTICAL BASIC PROGRAMS

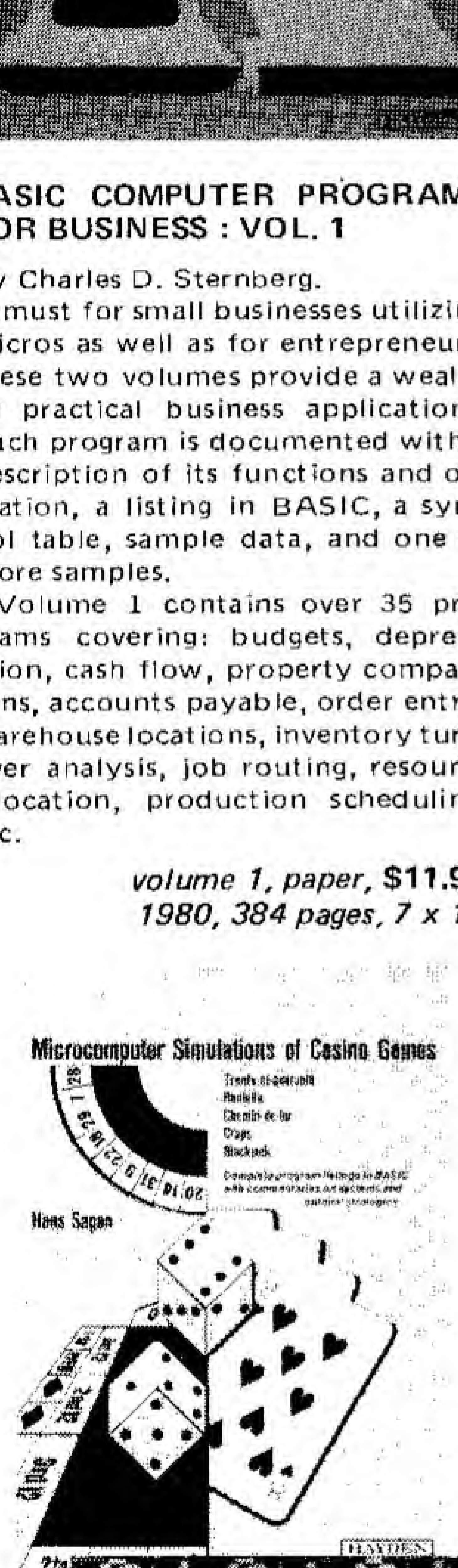
Edited by Lon Poole
Here is a new collection of 40 programs you can easily key in and use on most microcomputers. Each program does something useful. *Practical BASIC Programs* is especially useful in small business applications. It solves problems in finance, management decision, mathematics and statistics. It requires no prior programming knowledge. Each program is thoroughly documented. The book contains sample runs, practical problems, BASIC source listings, and an easy to follow narrative to help you realize the potential uses of each program. This book is a valuable reference for anyone who needs a wide range of useful programs: income averaging, present value of a tax shield, lease/buy decision, financial statement ratio analysis, checkbook reconciliation, home budgeting, nonlinear breakeven analysis, Program Evaluation and Review Technique (PERT), statistics, data forecasting divergence, musical transposition, Bayesian decision analysis, etc.

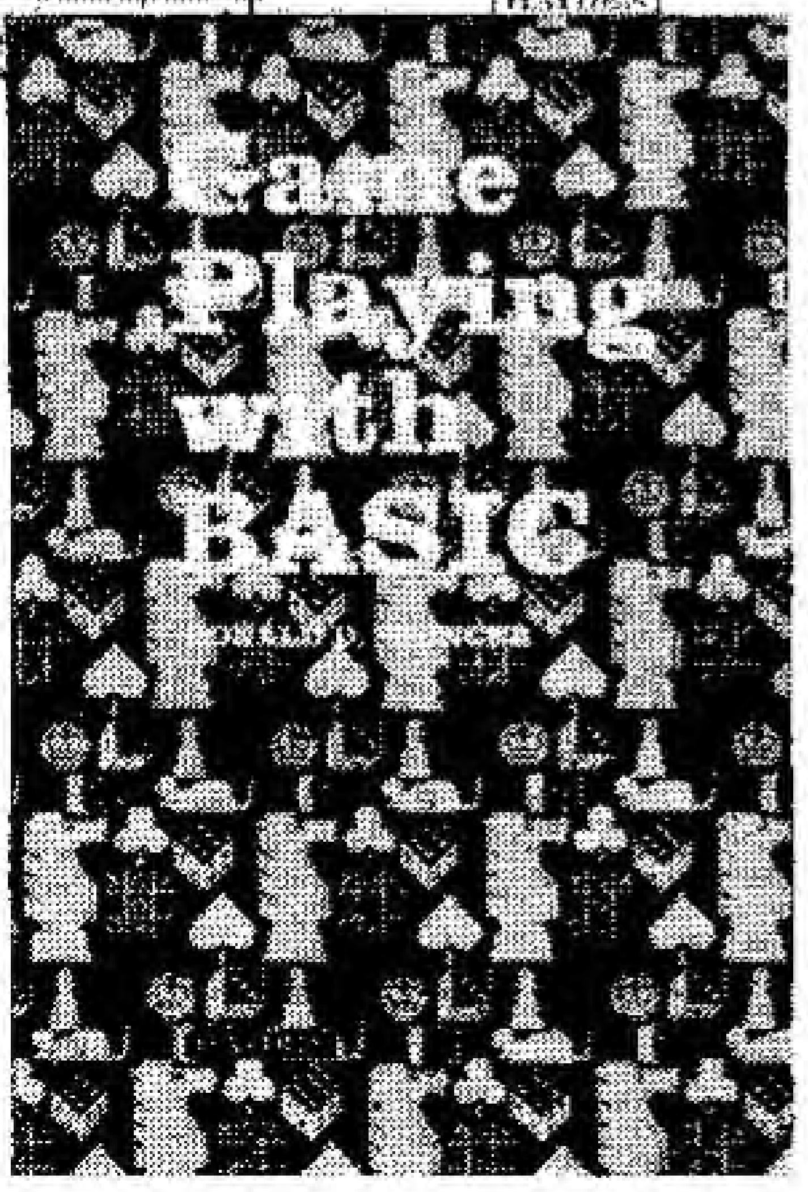
*paper,***$15.99**
*1980, 200 pages, 8½ x 11*

## BASIC COMPUTER PROGRAMS FOR THE HOME

By Charles D. Sternberg.
An invaluable book containing over 75 practical home application programs that will be helpful to the novice or experienced owner in increasing the usefulness of any home computer. Each program is documented with a description of its functions and operation, a listing in BASIC, a symbol table, sample data, and one or more samples. Programs included are: Home Financial Programs; Automobile Related Programs; Kitchen Helpmates; Scheduling Programs for Home Use; List Programs for Every Purpose; Miscellaneous Programs for the Home; Tutorial Programs for Home Use; Conversion Program; and Hobbyist's Diaries.

*paper,* **$10.95**
*1979, 336 pages, 7 x 10*





## BEAT THE ODDS: MICROCOMPUTER SIMULATIONS OF CASINO GAMES

By Hans Sagan.
Here's an extremely useful programming guide that provides realistic simulations of five popular Casino games: Trente-et-Quarante (Thirty and Forty), Roulette, Chemin-de-Fer, Craps, and Blackjack. Each of the five chapters has the same structure. It begins with a computer run, displaying facets of the programs, followed by an explanation of the objectives and the physical execution of the game. Acceptable bets and how to place them are discussed and systems and/or strategies laid out. Finally, the computer program is developed and various modifications of the program are detailed.

All programs are written in BASIC and heavily REM'd for readability and conversion. A comprehensive bibliography, a glossary of French gambling terms and phrases, and hints on the discrepancies between BASIC dialects are included, as well as a summary of maxims of probability theory.

*paper,* **$8.95**
*1980, 128 pages, 6 x 9*

## GAME PLAYING WITH BASIC

By Donald D. Spencer.
Enjoy the challenge of competition with your computer. Amuse yourself with such games and puzzles as 3-D Tic-tac-toe, Nim, Roulette, Magic Squares, the 15 Puzzle, Baccarat, Knight's Magic Tour, and many others. The writing is nontechnical, allowing almost anyone to understand computerized game playing. The book includes the rules of each game, how each game works, illustrative flowcharts, diagrams, and the output produced by each program. The last chapter contains 26 games for reader solution.

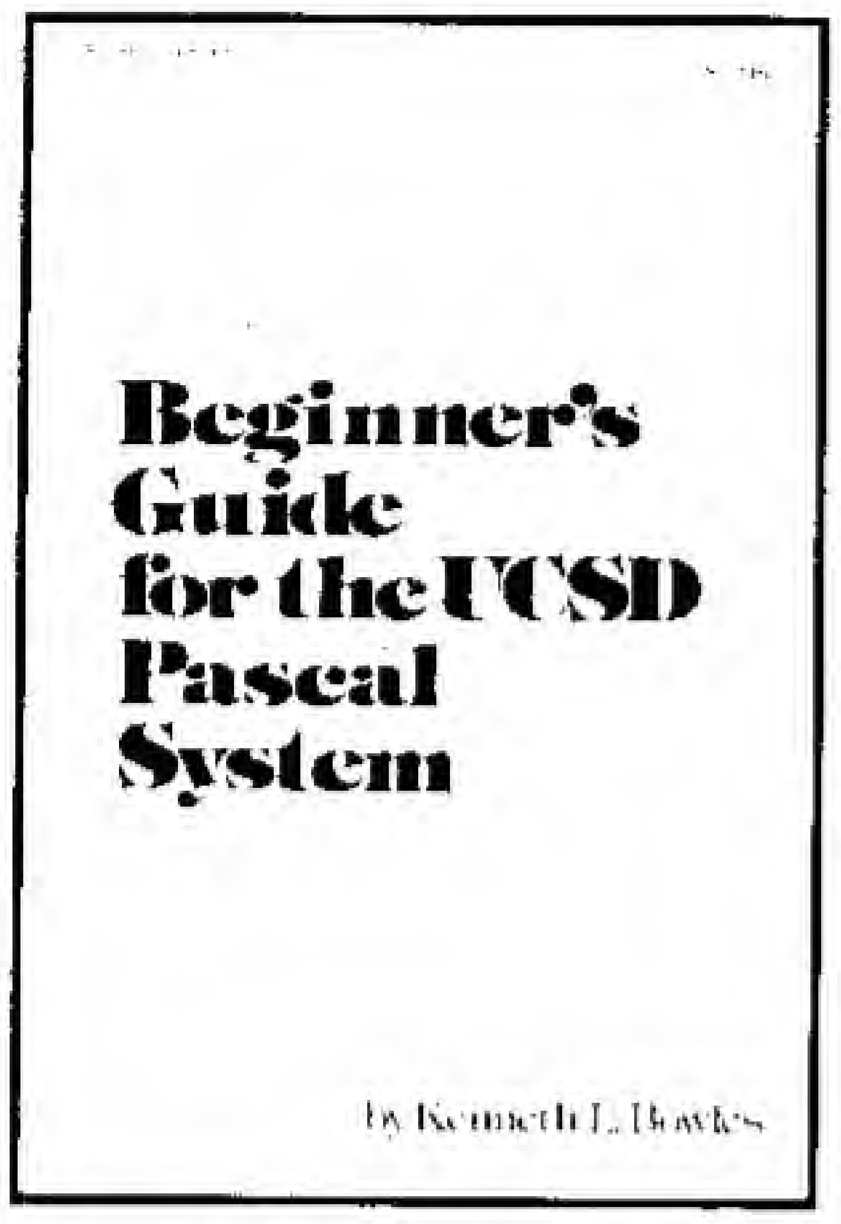
*paper,* **$10.50**
*1977, 176 pages, 6 x 9, illus.*

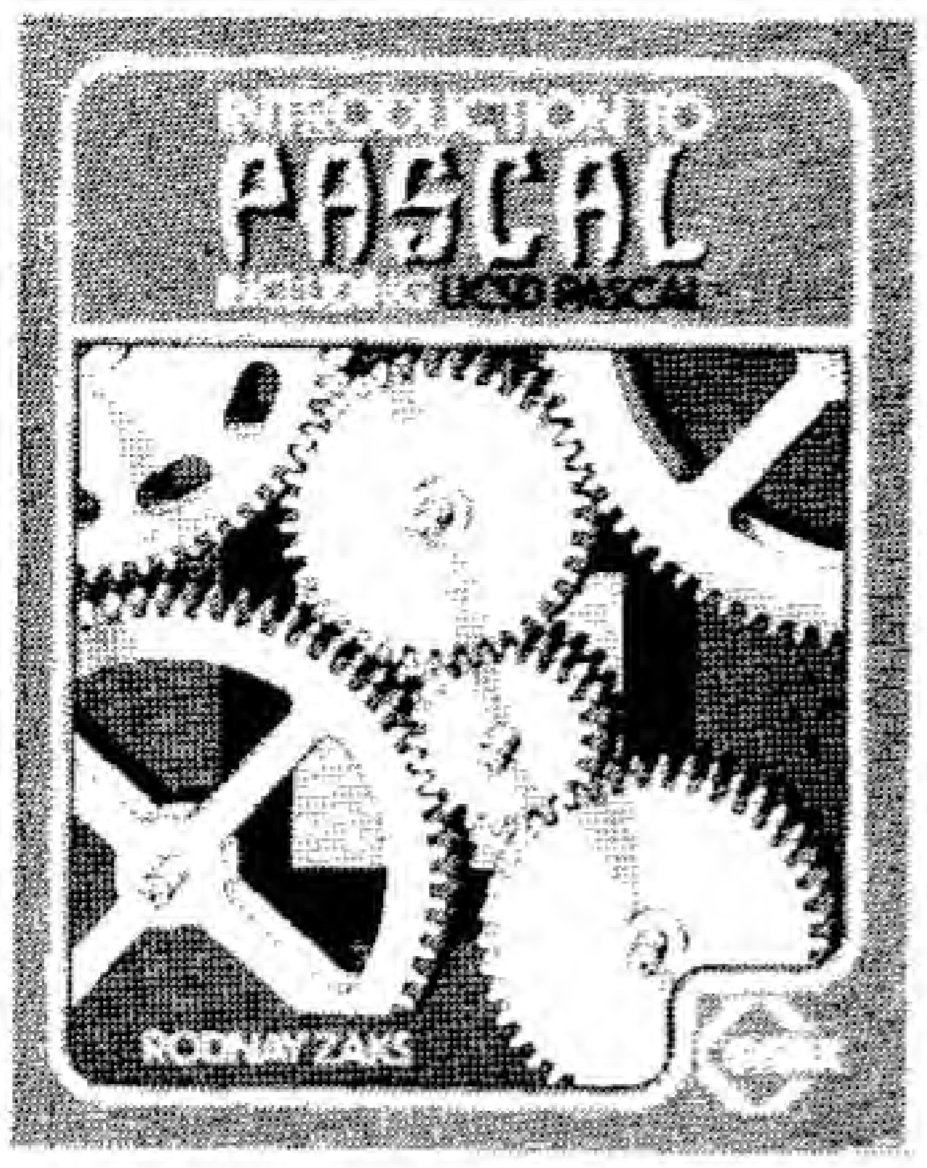
## MINDSTORMS: CHILDREN, COMPUTERS AND POWERFUL IDEAS

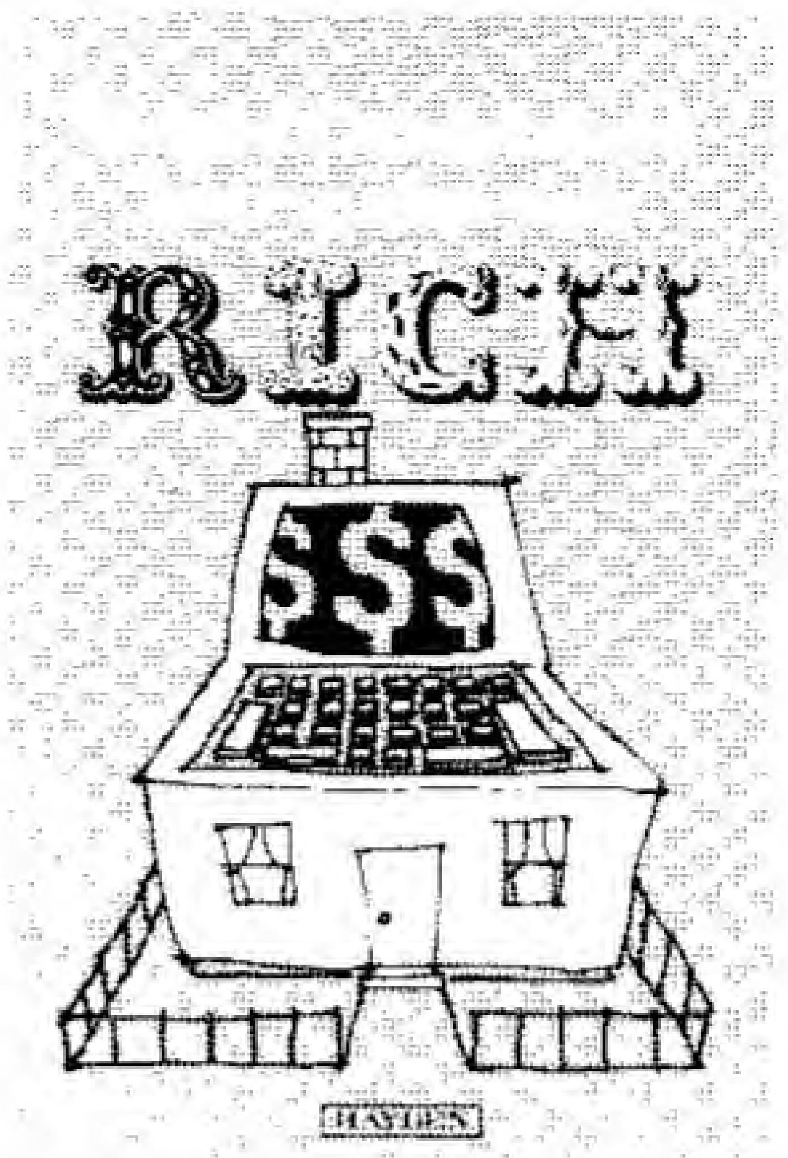
By Seymour Papert
The definitive work on the philosophy behind LOGO. Excerpted in the May/June issue of this magazine.

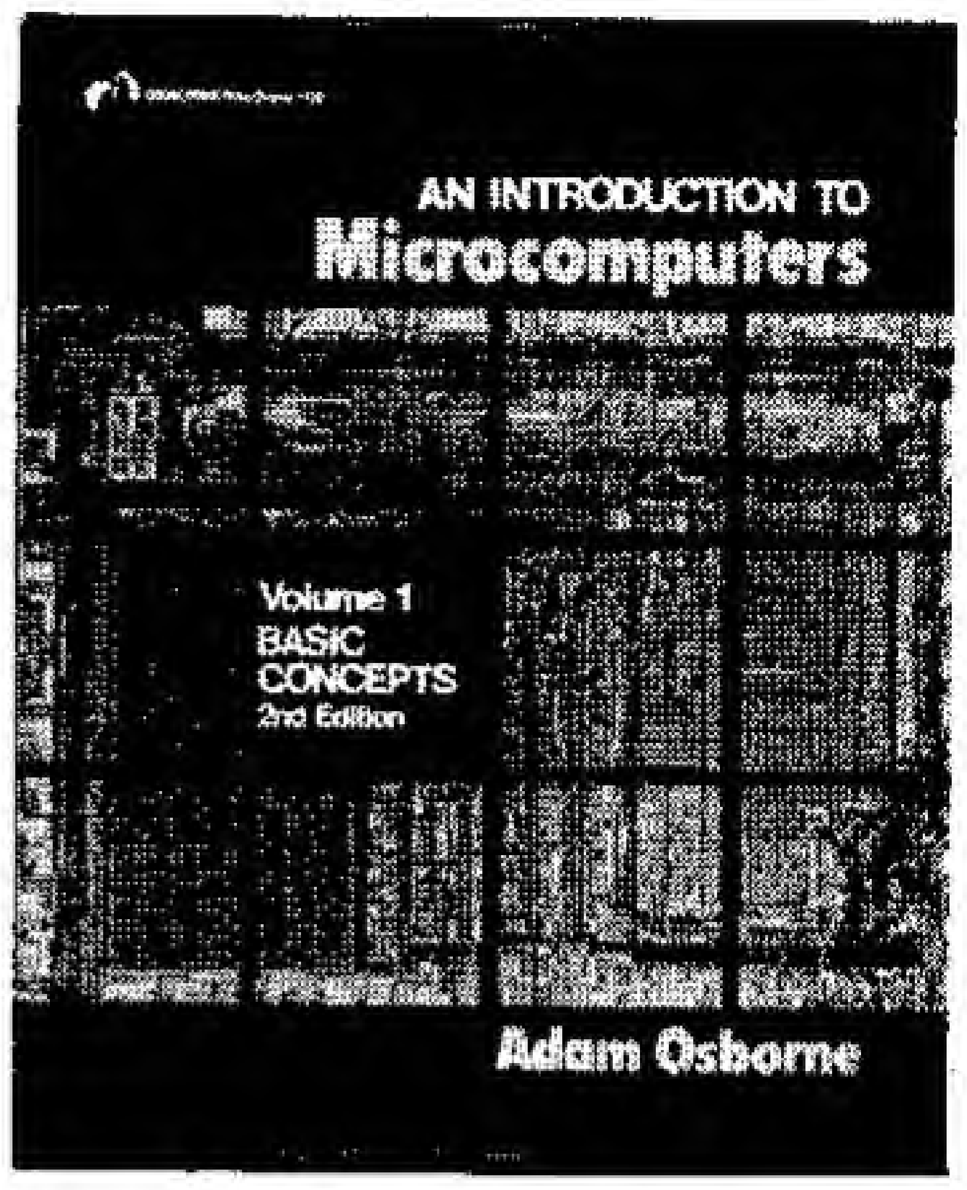
*Hardcover,* **$12.95**
*1980, 230 pages, 6 x 9*

# 99'er BOOKSTORE








AN INTRODUCTION TO Microcomputers
Volume 1 BASIC CONCEPTS 2nd Edition
Adam Osborne

**BEGINNER'S GUIDE FOR THE UCSD PASCAL SYSTEM**

By Kenneth Bowles
This highly informative book is written by the originator of the UCSD Pascal System. It is designed as an orientation guide for learning to use the UCSD Pascal System, and features tutorial examples of programming tasks in the form of self-study quiz programs. Once familiar with the system you will find the guide an invaluable reference tool for creating advanced applications.

*paper,* $11,95
*1980, 204 pages, 6 x 9*

Increase Your Knowledge. Read A Book Today !

**INTRODUCTION TO TI BASIC**

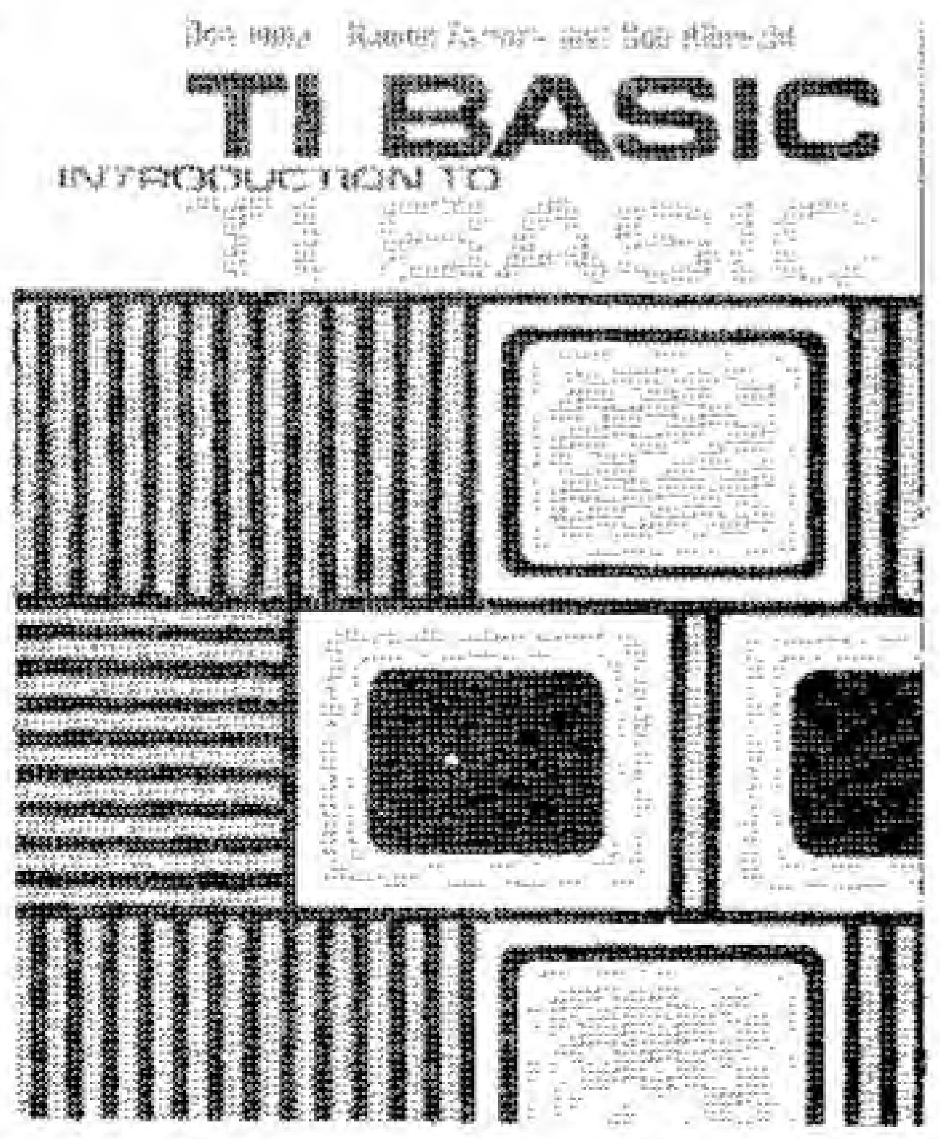
By D. Inman, R. Zamora, and R. Albrecht.
This comprehensive work will teach you all about computers and BASIC for use with the Texas Instruments Home Computer. Even if you've never worked with a computer, you can now teach yourself how to use, program and enjoy the TI Home Computer with this entertaining, and easy-to-read work. The authors have carefully constructed this introduction so that you will soon be writing BASIC programs and exploiting all of the excellent features of the TI machines. Its 14 chapters and Appendices cover all of the essential programming statements and machine features.

*paper,* $11.95
*1980, 320 pages, 7 x 10*

**INTRODUCTION TO PASCAL (INCLUDING UCSD PASCAL)**

By Rodnay Zaks
This is the first book on Pascal that can be used by persons who have never programmed before, but more generally it is a simple and comprehensive introduction to standard and UCSD Pascal for anyone—beginner to experienced programmer—who wants to learn the language rapidly. The logical progression and graduated exercises—designed to provide practice as well as test skill and comprehension—enable the reader to begin writing simple programs almost immediately. This book presents all concepts and techniques in a clear and simple style, making it accessible to beginners and useful to experienced programmers. All Pascal features are covered in detail, from basic definitions to complex data structures. An extensive appendix section presents a listing of all symbols, keywords and rules of syntax for programming in Pascal, providing a concise summary and important reference tool.

*paper,* $14.95
*1981, 440 pages, 7 x 9*

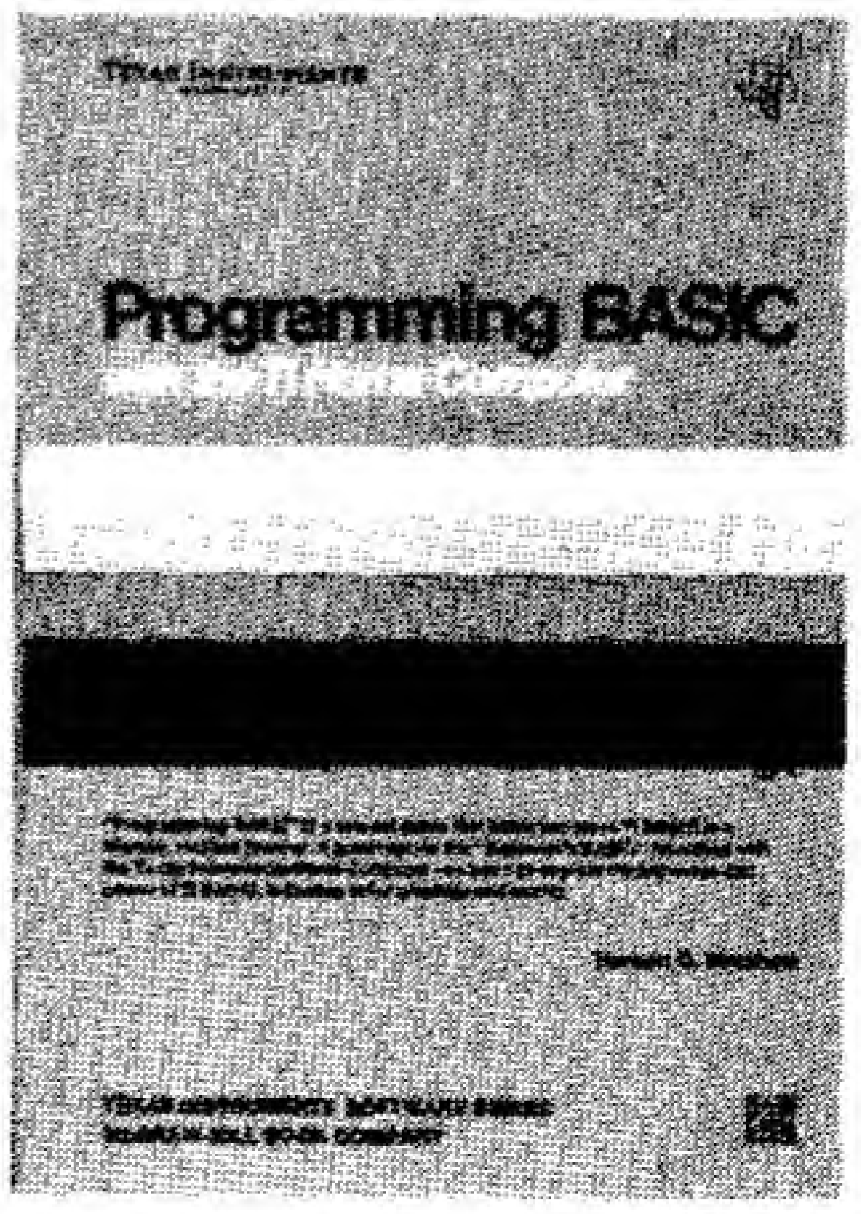


**HOME COMPUTERS CAN MAKE YOU RICH**

By Joe Weisbecker.
Here's a valuable text for every home computer owner and non-owner interested in spare-time income opportunities. You'll be introduced to the microcomputer industry, and the types of people involved in it. You'll find out how to learn more about this new industry. Discussed are basic principles of making money, freelance writing, programming, consulting, inventing, computer-made products, investing, and much more.

*CONTENTS:* The Microcomputer Industry. What you Need to Know About Making Money. Resources You Can Use. Choosing Your Hardware. Writing for Money. Creating and Selling Programs. Services for Sale. Use Your Imagination. Invent Your Way to Success. Making Your Money Grow. Working at Home.

*paper,* $7.50
*1980, 128 pages, 6 x 9*



**AN INTRODUCTION TO MICROCOMPUTERS – VOLUME 1 – BASIC CONCEPTS**

By Adam Osborne
Using concepts that are common to all microprocessor systems, *Volume 1* develops a detailed picture of what a microcomputer can do, how it does what it does, and how the capabilities of microcomputers can best be applied. Basic Concepts presents the fundamental logic framework upon which microcomputer systems are built, so that the reader can evaluate the applicability of microcomputers to any practical problem. This new revised edition incorporates all recent microprocessor developments. Concepts are discussed in terms of modern hardware configurations, and examples of common applications are drawn from today's most popular devices. For example, the logic instructions and programming concepts of the new 16-bit microprocessors are discussed in detail, and current logic distribution configurations are used throughout the text with numerous illustrations and examples. Programming mnemonics conform to the newly proposed IEEE standard. This is the first book in print to use them.

*paper,* $12.99
*1980, 320 pages, 7 x 9*

**PROGRAMMING BASIC WITH THE TI HOME COMPUTER**

By Herbert D. Peckham.
A tutorial guide that helps you learn TI BASIC in a friendly, relaxed manner. It goes beyond *Beginner's BASIC* furnished with the TI-99/4, and introduces the full range of TI BASIC features including color graphics and sound. Its 11 chapters are written in a complete-the-blanks, programmed instruction format.

*paper,* $14.50
*1979, 306 pages, 6 x 9*

**Use the order card in the back of this magazine, or itemize your order on a separate piece of paper and mail to: 99'er Magazine/Book Dept., 2715 Terrace View Drive, Eugene, OR 97405. Be sure to include check or detailed credit card information. No. C.O.D. orders accepted. Add $1.50 postage & handling for 1 book, $2.00 for 2 books, or $2.50 for 3 or more books. Please allow 4-6 weeks for delivery. If there is a question regarding your order please write to Customer Service at the above address. PRICES SUBJECT TO CHANGE WITHOUT NOTICE.**

# RECORDER ROSTRUM

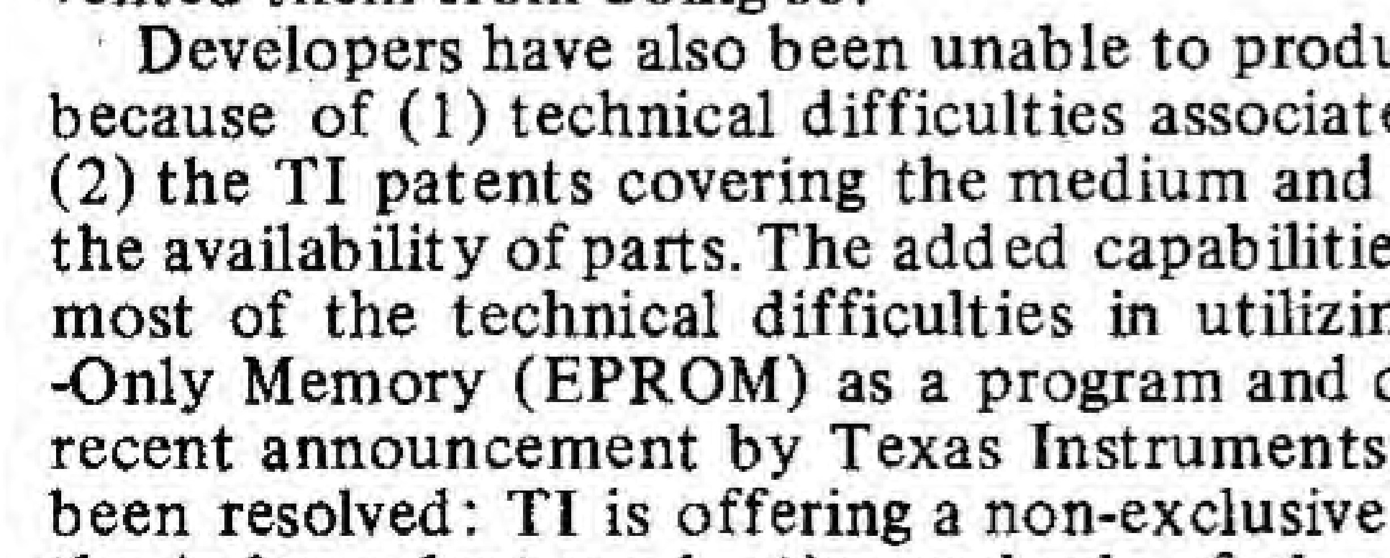
**Using Cassette Recorders with the TI-99/4 and TI-99/4A**

**By F.O. Armbruster**

One of the most frustrating things for a beginning computer user is being unable to find the optimum adjustment on a cassette recorder in order to reliably save a program on tape. It took me about two weeks of experimentation before I finally got it all figured out so that I could do it consistently without a hitch. Compounding the difficulty is the fact that some tape recorders have their motors wired up with polarity reversed from that of the computer. In this case an adapter is needed if the computer is to control the tape drive motor. When 99'er-ware introduced the TI SETTE adapter (pronounced "tie-set"), that problem was neatly solved. Since I thought it would be a great help to beginning users if better information on cassette recorders were available, I started testing the various brands and models of cassette recorders for compatibility with my TI-99/4. The results to date are shown in the accompanying chart. If any readers have information to add about a particular make and model, please communicate it to me in care of *99'er Magazine* and I will try to keep the listing up to date.

| Name Model | Requires TI-SETTE* | Comments, Features | Price Range |
|---|---|---|---|
| G.E. 3-5361 | no | small, nice design, works fine, tone cont. | $40-50 |
| G.E. 3-5105 F | no | older model discontinued, works fine | $50-60 |
| G.E. 3-5104 C | yes | no tone control, touchy on playback volume setting. | $25-35 |
| G.E. 3-5151 A | yes | has tone control, works fine | $40-50 |
| G.E. 3-5307 | yes | compact, hard to connect, works okay | $60-70 |
| G.E. 5154 A | no | also known as Slimline S-2, works fine, not avail. in some areas | $30-40 |
| G.E. 5005 B | yes | no tone control, touchy on volume control | $25-35 |
| Panasonic RQ335 | no | touchy on volume control, otherwise okay very compact, good for travel | $50-60 |
| Panasonic RQ337 | no | has dual sensitivity on mic., no tone control, works okay | $50-60 |
| Panasonic RQ 309AS | no | touchy on volume setting, otherwise okay | $40-50 |
| J.C. Penny # 681-3246 | no | best tested, very forgiving of volume setting errors | $40-50 |
| Realistic CTR-55 | no | no tone control, works okay | $20-30 |
| Realistic CTR-37 | no | no tone control, works okay | $30-40 |
| Craig J100 | yes | no tone control, works okay | $30-40 |
| Craig J103 | yes | small, tone control, nice design, works fine | $70-80 |

* See inside rear cover of this Magazine.

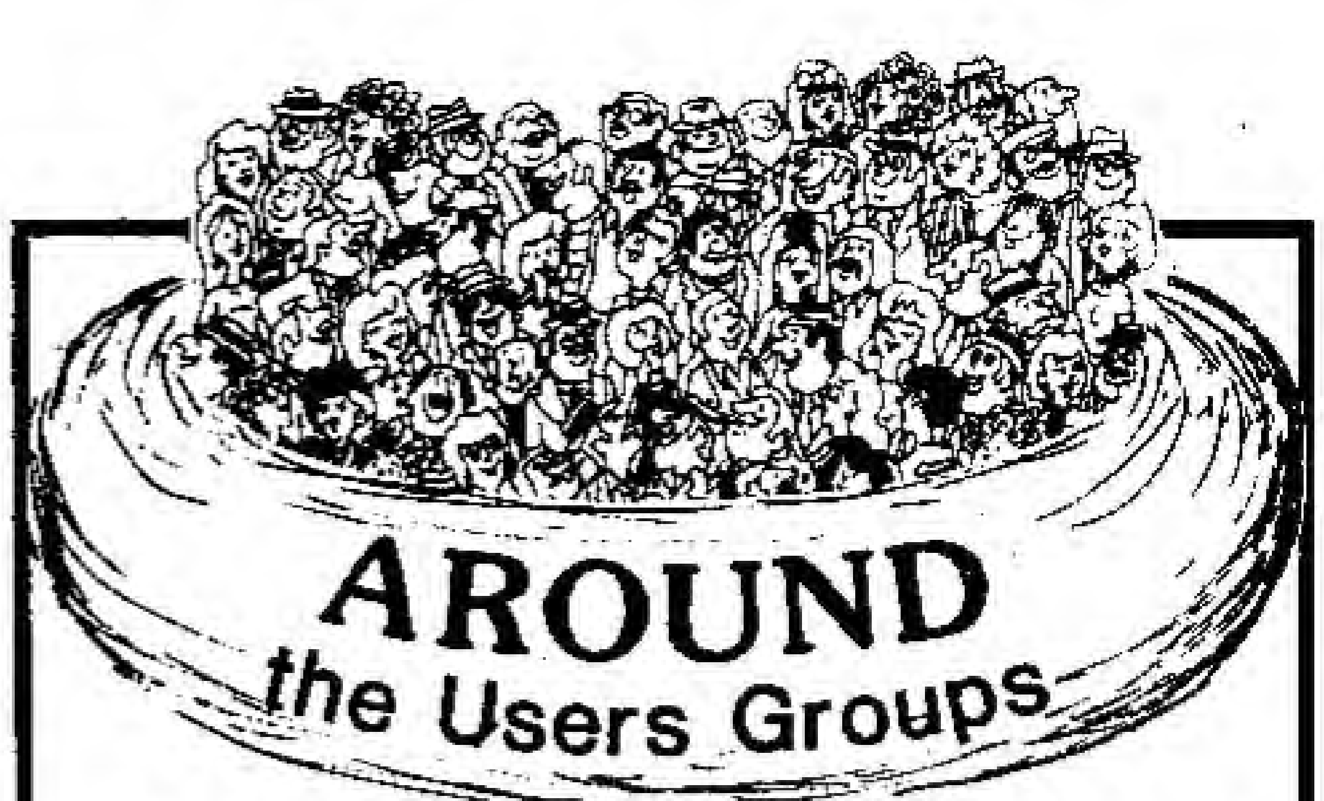


## 3rd-Party NEWS

**For Software Developers**

Many third-party firms have been interested in developing and marketing programs in the form of Solid State Software™ Command Modules, but the costs, timing, and minimum quantities for the TI-produced media have prevented them from doing so.

Developers have also been unable to produce their *own* Command Modules because of (1) technical difficulties associated with the old TI-99/4 console, (2) the TI patents covering the medium and its associated technology and (3) the availability of parts. The added capabilities of the TI-99/4A have alleviated most of the technical difficulties in utilizing Eraseable Programmable Read-Only Memory (EPROM) as a program and data medium. And as a result of a recent announcement by Texas Instruments, the legal problem has also now been resolved: TI is offering a non-exclusive royalty-bearing license for use in the independent production and sale of plug-in software for home computers. A copy of the standard terms and conditions is available upon request from: Manager, Third Party Software, Texas Instruments Inc., P. O. Box 10508, M/S 5890, Lubbock, TX 79408. Watch this column for an expected announcement of parts availability.

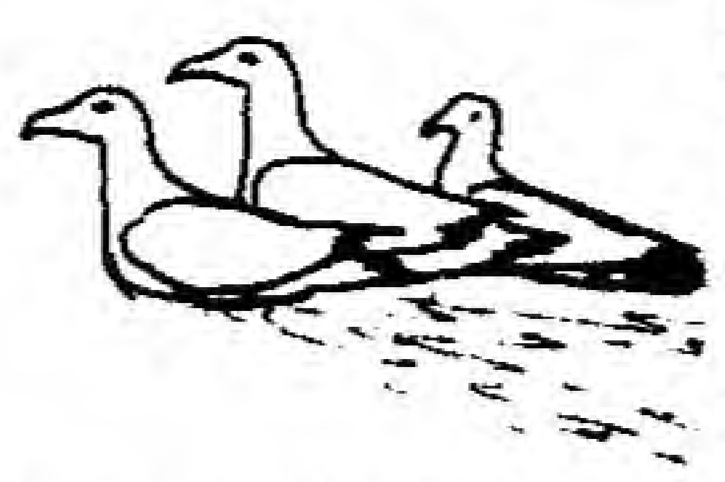


## AROUND the Users Groups

Spectacular graphics demonstrations with the new 9900 assembler have been taking place at a couple of the Texas groups: **Joe De Vecentis** demonstrated the sophisticated line drawing capabilities of the TI-99/4 to fellow **Houston** meeting attendees, while **Bernie Asner** showed the **Dallas** members a collection of super-fast techniques that left many of them wondering if this was the same computer they've been using all this time . . . The **Saudi Arabia** TI Rumor Exchange (SATIRE) is alive and well with **Jay Meeuwig** coordinating the 15-member group composed of Canadian and American oil drillers. **Jim Muller** of the **Young Peoples' LOGO Association** has started the wheels rolling on an "Earn a Computer" incentive program whereby the graphics programs submitted by those 18 years of age and younger are judged on originality and creativity, and balanced against the capabilities of the language and system used, plus the age/experience of the programmer. **Mark Moseley**, an active **Pacific Northwest** member and recipient of a new disk drive, has started churning out programs by the dozen. At a recent **Cincinnati-Dayton** meeting, **Don Owens** demonstrated and explained the development process for his new game program, "Farkle." **Charles LaFara** of the **International 99/4 Users-Group** was at the Consumer Electronics Show demonstrating the easy access to hundreds of group programs available for downloading via TEXNET's TI-XCHANGE. Also attending CES was **Larry De Rusha**, President of the **International Home Computer Users' Association**—a new multi-service organization that has been organized to provide users' groups and individual users of the Texas Instruments Home Computer with an information & referral service, consumer aid, a newsletter & bulletin, audiovisual support material for groups, and a software exchange. **Charles Ehninger** of Futura Software was the visiting guest speaker at a recent group meeting in **Houston**. A new **Dutch** group (Gebruikers-Groep TI 99/4 Nederland) sporting over 40 members and an 18-page newsletter is now officially in operation, with **Paul Karis** its first President. And in a gesture of international support, the **English** group (T.I. HOME) under the helm of **Paul Michael Dicks** has offerd the Dutch group their complete program library to get started. **Dick Bies** has been actively demonstrating to fellow **Pittsburgh** group members what can be done with TI LOGO. Members of the **Pacific Northwest** group had occasion to view the carpentry work of **John Goodson** when he displayed his custom TI-99/4 cabinet. "Programming Structure" was the topic of the well-received **Cincinnati-Dayton** seminar that was given by **Jim Schwaller**. There's some disappointment "down under" over the delayed introduction of the new TI-99/4A, but the **Australian** users are keeping active with their 99/4s—especially **Bernie Elsner** of Perth (on the West Coast) and **Alwyn Smith** of Brisbane (on the East Coast) who just held an historic transcontinental CHAT via satellite, TELENET, and The Source. **Shane Andersen**, the national coordinator for all Aussie groups, has been responsible for the Sydney organization receiving a lot recent publicity and acclaim.

**Please keep your group news coming . . .**

## Counting Lesson . . . from p. 45

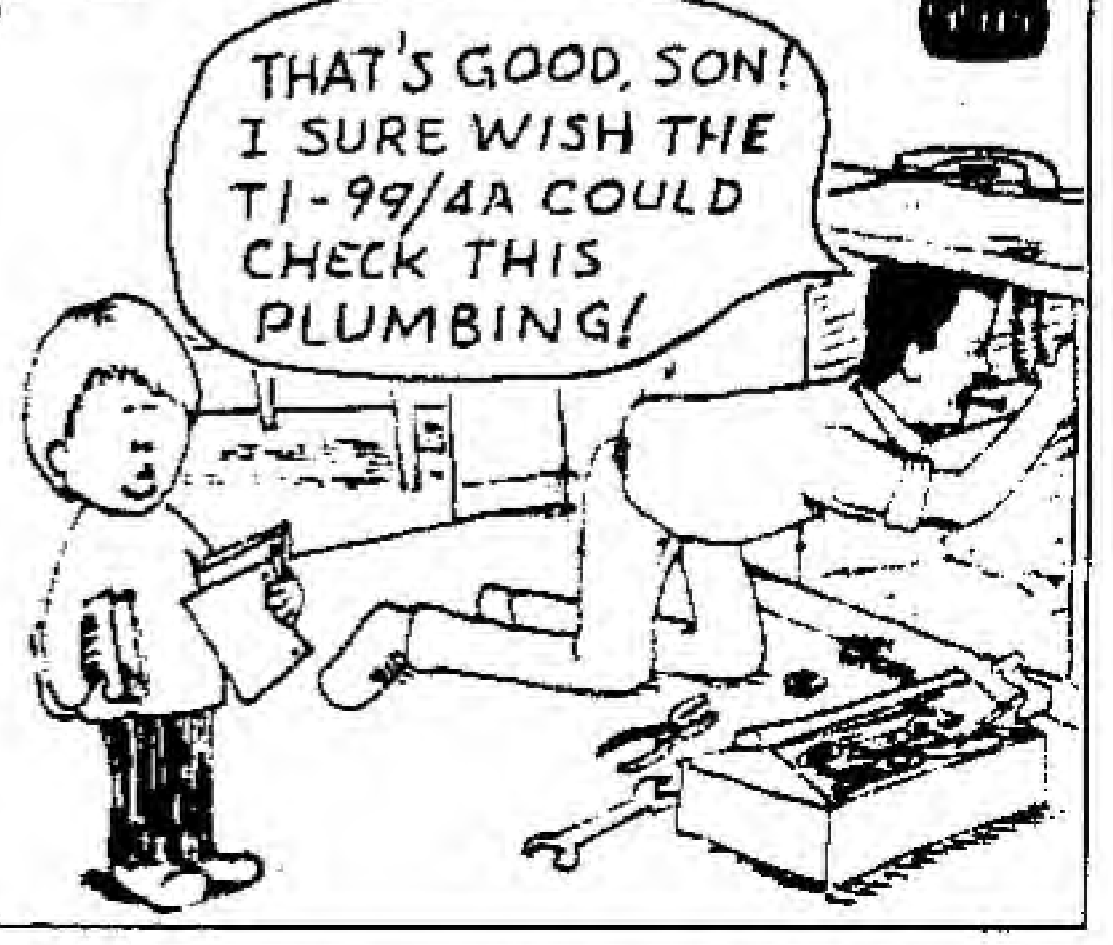


```
200 P(LOOP)=0
210 NEXT LOOP
220 T=0
230 DN=0
240 PN=0
250 L=0
260 N=0
270 CT=0
280 NM=0
290 REM  MENUS
300 PRINT "    INSTRUCTOR PLEASE NOTE"
310 PRINT "****************************"::
320 PRINT "DURING THE LESSON":"YOU CAN PRESS <AID> TO GET":"A READOUT OF PERCENT
    OF":"CORRECT RESPONSES"
330 PRINT :"IF YOU WISH TO RESET PROGRAM":"OPTIONS PRESS THE <BACK> KEY"::
340 PRINT :"DO YOU WANT TO USE SPEECH":"OPTION? (Y/N) (SPEECH EDITOR":"AND
    SYNTHESIZER REQUIRED)"
350 CALL KEY(0,K,S)
360 IF S=0 THEN 350
370 IF K=89 THEN 430
380 IF K<>78 THEN 350
390 SP=1
400 PN=1
410 CT=1
420 GOTO 440
430 SP=0
440 PRINT :::::"  SELECT FROM THE FOLLOWING":"****************************"::
450 PRINT TAB(8);"1-RANDOM GROUPS":TAB(8);"2-SEQUENTIAL":TAB(10);"PRESENTATION"
460 PRINT TAB(8);"3-END LESSON"
470 CALL KEY(0,K,S)
480 IF S=0 THEN 470
490 IF K=50 THEN 530
500 IF K=51 THEN 3070
510 IF K<>49 THEN 470
520 L=1
530 PRINT :"DO YOU WANT TO DISPLAY THE":"NUMBERS ABOVE THE GULLS?":"(Y/N)"::
540 CALL KEY(0,K,S)
550 IF S=0 THEN 540
560 IF K=89 THEN 590
570 IF K<>78 THEN 540
580 DN=1
590 IF SP=1 THEN 720
600 PRINT "DO YOU WANT TO PRONOUNCE":"EACH NUMBER AS IT IS PRINT-":"ED?
    (Y/N)"::
610 CALL KEY(0,K,S)
620 IF S=0 THEN 610
630 IF K=89 THEN 660
640 IF K<>78 THEN 610
650 PN=1
660 PRINT "DO YOU WANT THE COMPUTER TO":"TELL THE STUDENT WHICH KEY":"TO PRESS
    FOR EACH TRIAL?":"(Y/N)"
670 CALL KEY(0,K,S)
680 IF S=0 THEN 670
690 IF K=89 THEN 720
700 IF K<>78 THEN 670
710 CT=1
720 PRINT ::"DO YOU WANT THE GULLS TO BE":"PRESENTED:":"    1-ALL ON SAME
    LINE":"    2-DIAGONAL":"    3-RANDOM"
730 CALL KEY(0,K,S)
740 IF S=0 THEN 730
750 IF K=49 THEN 820
760 IF K=50 THEN 800
770 IF K<>51 THEN 730
780 LN=2
790 GOTO 840
800 LN=1
810 GOTO 840
820 LN=0
830 REM  DEFINE CHARACTERS
840 RANDOMIZE
850 CALL CLEAR
860 CALL SCREEN(8)
870 A$="E0F0F8FCFEFFFFFB"
880 B$="7F3F1F0F070FFFFF"
890 C$="00000000000080C0"
900 D$="E0F0F8FCFEFFFFFF"
910 E$=""
920 F$="0103070F1F3F7FFF"
930 G$="000101010103070F"
940 H$="808080F0F8FCFEFF"
950 I$="FFF7FFBAFFBFEBFF"
960 J$="3337F7FF5FFAFFFF"
970 K$="000000003F1F0F07"
980 L$="FFAAFFFFFFFFFFFF"
990 M$="000000FFFFFFFEFC"
1000 N$="00000000C3C3C3C3"
1010 O$="0000000000081422"
1020 P$="000E0F0C86FE7E3C"
1030 Q$=""
1040 R$="418200C300C3C3C3"
1050 S$=""
1060 CALL CHAR(96,A$)
1070 CALL CHAR(97,B$)
1080 CALL CHAR(98,C$)
1090 CALL CHAR(99,D$)
1100 CALL CHAR(104,E$)
1110 CALL CHAR(100,F$)
1120 CALL CHAR(101,G$)
1130 CALL CHAR(112,H$)
1140 CALL CHAR(113,I$)
1150 CALL CHAR(114,J$)
1160 CALL CHAR(120,K$)
1170 CALL CHAR(121,L$)
1180 CALL CHAR(122,M$)
1190 CALL CHAR(128,N$)
1200 CALL CHAR(136,O$)
1210 CALL CHAR(102,P$)
1220 CALL CHAR(140,Q$)
1230 CALL CHAR(144,R$)
1240 CALL CHAR(145,S$)
1250 CALL COLOR(14,8,6)
1260 CALL COLOR(9,16,6)
1270 CALL COLOR(11,13,11)
1280 CALL COLOR(10,11,11)
1290 CALL COLOR(12,7,11)
1300 CALL COLOR(13,2,11)
1310 CALL COLOR(15,16,11)
1320 CALL HCHAR(1,1,104,288)
1330 CALL HCHAR(10,1,140,480)
1340 CALL HCHAR(9,10,112)
1350 CALL HCHAR(9,1,113,9)
1360 CALL HCHAR(8,9,112)
1370 CALL HCHAR(8,1,113,8)
1380 CALL HCHAR(7,8,112)
1390 CALL HCHAR(7,1,113,7)
1400 CALL HCHAR(6,7,112)
1410 CALL HCHAR(6,1,114,6)
1420 CALL HCHAR(9,27-X,120)
1430 CALL HCHAR(9,28-X,121,3)
1440 CALL HCHAR(9,31-X,122)
1450 CALL HCHAR(8,29-X,128)
1460 REM   CALCULATE NUMBER OF GULLS
1470 IF L<>0 THEN 1540
1480 IF T=1 THEN 1540
1490 N=N+1
1500 IF N>9 THEN 1540
1510 GOTO 1570
1520 N=1
1530 GOTO 1570
1540 N=INT(RND*9)+1
1550 T=1
1560 REM  CLEAR 'WATER' AREA
1570 FOR ERASE=10 TO 24
1580 CALL HCHAR(ERASE,1,140,32)
1590 NEXT ERASE
1600 REM   LOOP PLACES GULLS IN
     WATER
1610 FOR J=1 TO (2*N)STEP 2
1620 NM=NM+1
1630 REM   CALCULATE POSITION OF
     GULLS
1640 IF LN=0 THEN 1690
1650 IF LN=1 THEN 1680
1660 P(NM)=(RND*8)+1
1670 GOTO 1690
1680 P(NM)=J/2
1690 CALL HCHAR(12+P(NM),J+9,136)
1700 IF DN=1 THEN 1720
1710 CALL HCHAR(11+P(NM),J+10,
     ASC(STR$(NM)))
1720 CALL HCHAR(12+P(NM),J+10,102)
1730 IF PN=1 THEN 1750
1740 CALL SAY(STR$(NM))
1750 NEXT J
1760 C=1
1770 CALL HCHAR(10,1,140,32)
1780 F$="B0C0E0F0F8FCFEFF"
1790 CALL CHAR(100,F$)
1800 IF CT=1 THEN 1840
1810 CALL SAY("PRESS")
1820 CALL SAY(STR$(NM))
1830 REM  START SHARK FIN ACROSS
     SCREEN IN FOREGROUND
1840 CALL HCHAR(22,C,100)
1850 CALL SOUND(25,110,0)
1860 CALL HCHAR(22,C,136)
1870 C=C+1
1880 FB=0
1890 IF C=32 THEN 2140
1900 CALL KEY(0,K,STATUS)
1910 IF STATUS=0 THEN 1840
1920 IF K=1 THEN 3080
1930 IF K=15 THEN 160
1940 IF K<48 THEN 2920
1950 IF K>57 THEN 2920
1960 NN=K-48
1970 IF NN<>NM THEN 2330
1980 NM=0
1990 REM  PLAY MUSICAL FLOURISH ON
     CORRECT RESPONSE
2000 CALL SOUND(100,349,0)
2010 CALL SOUND(100,440,0)
```

Continued on p. 84

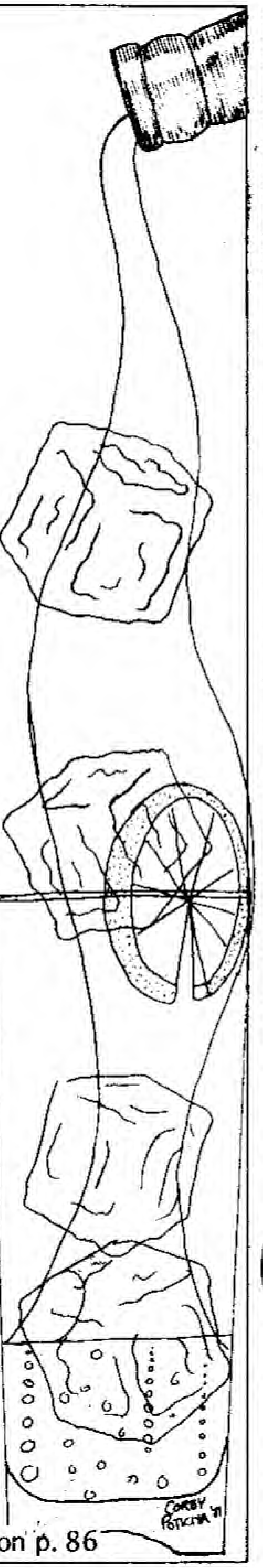
## Homework Helper . . . from p. 43

```
700 CALL COLOR(2,16,7)
710 CALL CHAR(126,"F8F8FCFCFEFEFFFF")
720 CALL CHAR(127,"FFFF7F7F3F3F1F1F")
730 CALL COLOR(2,7,16)
740 CALL CHAR(128,"FFFFFEFEFCFCF8F8")
750 CALL CHAR(129,"0F0F0703")
760 CALL COLOR(2,16,7)
770 CALL CHAR(130,"FFFFFFFFFF3F0F03")
780 CALL CHAR(131,"FFFFFFFFFFFCF0C")
790 CALL COLOR(2,7,16)
800 CALL CHAR(132,"F0F0E0C")
810 CALL COLOR(2,2,1)
820 CALL CLEAR
830 CALL CHAR(120,"FFFFFFFFFFFFFFFF")
840 CALL COLOR(12,1,1)
850 CALL COLOR(13,1,1)
860 CALL SOUND(T/2,262,5)
870 CALL VCHAR(15,6,120,5)
880 CALL SOUND(T/2,294,5)
890 CALL HCHAR(15,4,121)
900 CALL SOUND(T,330,4,131,10)
910 CALL HCHAR(15,5,122)
920 CALL HCHAR(15,7,123)
930 CALL HCHAR(15,8,124)
940 CALL SOUND(T,330,4,196,10)
950 CALL HCHAR(16,4,125)
960 CALL HCHAR(16,5,120,3)
970 CALL HCHAR(16,8,126)
980 CALL SOUND(T/2,330,4)
990 CALL HCHAR(17,4,120,5)
1000 CALL SOUND(T/2,392,4)
1010 CALL HCHAR(18,4,127)
1020 CALL SOUND(T,349,3,147,9)
1030 CALL HCHAR(18,5,120,3)
1040 CALL HCHAR(18,8,128)
1050 CALL HCHAR(19,4,129)
1060 CALL HCHAR(19,5,130)
1070 CALL SOUND(T,349,3,196,9)
1080 CALL HCHAR(19,7,131)
1090 CALL HCHAR(19,8,132)
1100 CALL COLOR(12,7,1)
1110 CALL COLOR(13,7,1)
1120 CALL SOUND(T/2,349,3)
1130 CALL COLOR(14,15,1)
1140 CALL COLOR(9,5,1)
1150 CALL SOUND(T/2,330,3)
1160 CALL HCHAR(13,9,136)
1170 CALL HCHAR(14,9,137)
1180 CALL SOUND(T,294,3,175,9)
1190 CALL HCHAR(14,10,136)
1200 CALL HCHAR(15,10,137)
1210 CALL HCHAR(15,11,136)
1220 CALL HCHAR(16,11,137)
1230 CALL SOUND(T/2,294,3,196,9)
1240 CALL HCHAR(16,12,136)
1250 CALL HCHAR(17,12,137)
1260 CALL SOUND(T/2,330,3)
1270 CALL HCHAR(17,13,136)
1280 CALL SOUND(T/2,349,2)
1290 CALL HCHAR(18,13,137)
1300 CALL SOUND(T/2,370,2)
1310 CALL HCHAR(18,14,136)
1320 CALL SOUND(T,392,1,165,7)
1330 CALL COLOR(10,7,1)
1340 CALL HCHAR(18,15,109)
1350 CALL HCHAR(19,14,110)
1360 CALL HCHAR(19,15,111)
1370 CALL SOUND(T,392,1,196,7)
1380 CALL HCHAR(20,16,111)
1390 CALL COLOR(11,5,1)
1400 CALL SOUND(T/2,440,2)
1410 CALL SOUND(T/2,494,2)
1420 CALL SOUND(T,523,1,165,7)
1430 CALL SOUND(T,523,1,196,7)
1440 CALL SOUND(T/2,587,2)
1450 CALL CHAR(120,"7FBFDFEFF7FBFDFE")
1460 CALL SOUND(T/2,523,2)
1470 CALL SOUND(T,494,1,175,7)
1480 CALL SOUND(T,784,1,196,7)
1490 CALL HCHAR(17,18,61)
1500 CALL SOUND(T/2,587,1)
1510 CALL SOUND(T/2,523,1)
1520 CALL SOUND(T,494,0,175,6)
1530 CALL HCHAR(13,22,108,7)
1540 CALL SOUND(T,880,0,196,6)
1550 CALL HCHAR(15,22,108,7)
1560 CALL HCHAR(17,22,108,7)
1570 CALL SOUND(T,494,0)
1580 CALL HCHAR(19,22,108,7)
1590 CALL HCHAR(21,22,108,4)
1600 CALL SOUND(T*2,523,1,165,7,131,9)
1610 CALL HCHAR(24,1,67)
1620 CALL HCHAR(24,2,72)
1630 PRINT "DOSE:"
1640 PRINT :"1 DIVISION WITH REMAINDER"
1650 PRINT :"2 DIVISION WITH DECIMAL"
1660 PRINT :"3 CONVERT FRACTION TO DECIMAL"
1670 CALL HCHAR(23,31,76)
1680 PRINT :"4 END PROGRAM"
1690 CALL KEY(0,K,S)
1700 IF (K<49)+(K>52)THEN 1690
1710 CALL CLEAR
1720 ON K-48 GOTO 1860,1990,2100,2360
1730 DATA 96,97,98,97,99,100,101,102,103,
     104,105,104,103,106,107,103,32
1740 DATA 112,113,114,115,116,117,118,115,32
1750 CALL COLOR(10,5,1)
1760 RESTORE 1730
1770 FOR I=7 TO 23
1780 READ G
1790 CALL HCHAR(19,I,G)
1800 NEXT I
1810 FOR I=15 TO 23
1820 READ G
1830 CALL HCHAR(18,I,G)
1840 NEXT I
1850 RETURN
1860 PRINT "DIVISION WITH REMAINDER"::::::::::
1870 GOSUB 1750
1880 CALL HCHAR(18,25,119)
1890 CALL HCHAR(18,26,35,2)
1900 INPUT "DIVISOR:   ":D
1910 IF D=0 THEN 820
1920 INPUT "DIVIDEND: ":N
1930 C=INT(N/D)
1940 R=N-C*D
1950 PRINT :"QUOTIENT =";C;" R";R
1960 PRINT :::"ENTER NEXT PROBLEM"
1970 PRINT "OR '0' TO STOP."::::::::::::
1980 GOTO 1870
1990 PRINT "DIVISION WITH DECIMAL"::::::::::
2000 GOSUB 1750
2010 CALL HCHAR(18,23,46)
2020 CALL HCHAR(18,24,35,3)
2030 INPUT "DIVISOR:   ":D
2040 IF D=0 THEN 820
2050 INPUT "DIVIDEND: ":N
2060 PRINT :"QUOTIENT =";N/D
2070 PRINT :::"ENTER NEXT PROBLEM"
2080 PRINT "OR '0' TO STOP."::::::::::::
2090 GOTO 2000
2100 PRINT "CONVERT FRACTION TO DECIMAL"::::::::::::
2110 CALL CHAR(120,"FFFFFFFFFFFFFFFF")
2120 CALL CHAR(138,"000000FFFF")
2130 CALL HCHAR(16,8,120,4)
2140 CALL HCHAR(17,8,120,4)
2150 CALL HCHAR(18,7,138,6)
2160 CALL HCHAR(19,8,120,4)
2170 CALL HCHAR(20,8,120,4)
2180 CALL HCHAR(21,8,120,4)
2190 CALL HCHAR(18,15,61)
2200 CALL HCHAR(18,18,46)
2210 CALL COLOR(10,13,13)
2220 CALL HCHAR(18,20,108)
2230 CALL HCHAR(18,22,108)
2240 CALL HCHAR(18,24,108)
2250 PRINT :::
2260 INPUT "NUMERATOR:    ":N
2270 IF N=0 THEN 820
2280 INPUT "DENOMINATOR: ":D
2290 IF D<>0 THEN 2320
2300 PRINT "SORRY, CANNOT DIVIDE BY ZERO"::
2310 GOTO 2280
2320 PRINT :N;"/";D;"=";N/D
2330 PRINT :::"ENTER NEXT PROBLEM"
2340 PRINT "OR ENTER '0' TO STOP."
2350 GOTO 2250
2360 END
```


THAT'S GOOD, SON! I SURE WISH THE TI-99/4A COULD CHECK THIS PLUMBING!

## Bartender . . . from p. 39

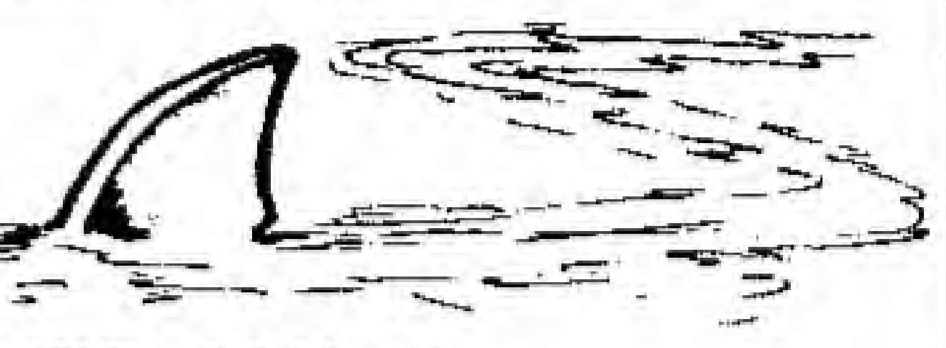
```
280 CALL COLOR(12,C,C)
290 RETURN
300 REM  NORMAL GLASS
310 CALL HCHAR(4,23,105,5)
320 CALL HCHAR(13,23,104,5)
330 CALL VCHAR(5,22,96,8)
340 CALL VCHAR(5,28,103,8)
350 CALL HCHAR(13,22,106)
360 CALL HCHAR(13,28,107)
370 CALL HCHAR(4,28,108)
380 CALL HCHAR(4,22,109)
390 FOR I=12 TO 7 STEP -1
400 FOR J=27 TO 23 STEP -1
410 CALL HCHAR(1,J,120)
420 NEXT J
430 NEXT I
440 RETURN
450 REM  COCKTAIL GLASS
460 CALL HCHAR(6,21,129)
470 FOR I=1 TO 3
480 CALL HCHAR(6+I,21+I,113)
490 NEXT I
500 FOR I=1 TO 3
510 CALL HCHAR(6+I,20+I,102)
520 NEXT I
530 CALL HCHAR(6,28,128)
540 FOR I=1 TO 3
550 CALL HCHAR(6+I,28-I,112)
560 NEXT I
570 FOR I=1 TO 3
580 CALL HCHAR(6+I,29-I,97)
590 NEXT I
600 CALL HCHAR(10,24,100)
610 CALL HCHAR(10,25,101)
620 CALL VCHAR(11,24,96,3)
630 CALL VCHAR(11,25,103,3)
640 CALL HCHAR(14,23,104,4)
650 CALL HCHAR(5,21,105,8)
660 CALL HCHAR(7,23,120,4)
670 CALL HCHAR(8,24,120,2)
680 RETURN
690 REM  TALL GLASS
700 CALL HCHAR(4,23,105,4)
710 CALL HCHAR(15,23,104,4)
720 CALL VCHAR(5,22,96,10)
730 CALL VCHAR(5,27,103,10)
740 CALL HCHAR(4,22,109)
750 CALL HCHAR(4,27,108)
760 CALL HCHAR(15,22,106)
770 CALL HCHAR(15,27,107)
780 FOR I=14 TO 7 STEP -1
790 FOR J=26 TO 23 STEP -1
800 CALL HCHAR(1,J,120)
810 NEXT J
820 NEXT I
830 RETURN
840 REM  OLD-FASH GLASS
850 CALL HCHAR(4,23,105,7)
860 CALL HCHAR(11,23,104,7)
870 CALL VCHAR(5,22,96,6)
880 CALL VCHAR(5,30,103,6)
890 CALL HCHAR(11,22,106)
900 CALL HCHAR(11,30,107)
910 CALL HCHAR(4,30,108)
920 CALL HCHAR(4,22,109)
930 FOR I=10 TO 7 STEP -1
940 FOR J=29 TO 23 STEP -1
950 CALL HCHAR(I,J,120)
960 NEXT J
970 NEXT I
980 RETURN
990 REM  PONY GLASS
1000 CALL VCHAR(5,22,96,5)
1010 CALL VCHAR(5,27,103,5)
1020 CALL HCHAR(9,26,112)
1030 CALL HCHAR(10,25,112)
1040 CALL HCHAR(10,26,97)
1050 CALL HCHAR(9,23,113)
1060 CALL HCHAR(10,24,113)
1070 CALL HCHAR(10,23,102)
1080 CALL HCHAR(11,24,100)
1090 CALL HCHAR(11,25,101)
1100 CALL VCHAR(12,24,96,3)
1110 CALL VCHAR(12,25,103,3)
1120 CALL HCHAR(15,23,104,4)
1130 CALL HCHAR(4,23,105,4)
1140 CALL HCHAR(4,22,109)
1150 CALL HCHAR(4,27,108)
1160 CALL HCHAR(9,24,120,2)
1170 CALL HCHAR(8,23,120,4)
1180 CALL HCHAR(7,23,120,4)
1190 RETURN
1200 REM  SPK OLIVE OR CHERRY
1210 CALL HCHAR(7,24,150)
1220 CALL HCHAR(6,23,98)
1230 RETURN
1240 REM  LEMON TWIST
1250 CALL HCHAR(7,24,152)
1260 CALL HCHAR(7,25,153)
1270 RETURN
1280 REM  ORANGE
1290 CALL COLOR(16,12,8)
1300 CALL HCHAR(5,22,157)
1310 CALL HCHAR(5,23,154)
1320 CALL HCHAR(6,22,155)
1330 CALL HCHAR(6,23,156)
1340 CALL HCHAR(4,21,98)
1350 RETURN
1360 CALL CHAR(152,"7F7F7F7F")
1370 CALL CHAR(110,"030303030303FFFF")
1380 CALL CHAR(158,"00000000C8CDFFFF")
1390 CALL CHAR(153,"FEFEFEFE")
1400 CALL CHAR(112,"0103070F1F3F7FFF")
1410 CALL CHAR(128,"0103070F1F3F7FFF")
1420 CALL CHAR(97,"FFFEFCF8F0E0C080")
1430 CALL CHAR(150,"3C7EFFFFFFFF7E3C")
1440 CALL CHAR(99,"2424242424242424")
1450 CALL CHAR(100,"FF7F3F1F0F070303")
1460 CALL CHAR(101,"FFFEFCF8F0E0C0C")
1470 CALL CHAR(102,"FF7F3F1F0F070301")
1480 CALL CHAR(113,"80C0E0F0F8FCFEFF")
1490 CALL CHAR(129,"80C0E0F0F8FCFEFF")
1500 CALL CHAR(106,"030303")
1510 CALL CHAR(107,"C0C0C0")
1520 CALL CHAR(108,"000000000000C0C")
1530 CALL CHAR(109,"0000000000000303")
1540 CALL CHAR(104,"FFFFFFFFFFFF")
1550 CALL CHAR(96,"0303030303030303")
1560 CALL CHAR(103,"C0C0C0C0C0C0C0C")
1570 CALL CHAR(105,"000000000000FFFF")
1580 GOSUB 460
1590 CALL CHAR(136,"3F7FFFFCFCFF7F3F")
1600 CALL CHAR(137,"FCFEFF3F3FFFFEFC")
1610 CALL CHAR(98,"8040201008040201")
1620 CALL CHAR(157,"0F0F3030C8C4C2C1")
1630 CALL CHAR(154,"F0F00C0C13234383")
1640 CALL CHAR(155,"C1C2C4C830300F0F")
1650 CALL CHAR(156,"834323130C0CF0F0")
1660 FOR I=1 TO 19
1670 CALL COLOR(2,16,7)
1680 FOR J=0 TO 23
1690 READ DRINK$(I,J)
1700 NEXT J
1710 CALL COLOR(2,7,16)
1720 NEXT I
1730 FOR I=0 TO 15
1740 READ INV$(I,0)
1750 NEXT I
1760 CALL CLEAR
1770 CALL COLOR(2,2,1)
1780 PRINT "DO YOU:":
     :"(1) WANT TO SEE THE RECIPE"
1790 PRINT "    FOR A SPECIFIC DRINK?"
1800 PRINT ::"(2) WANT TO KNOW WHAT YOU"
1810 PRINT "    CAN MAKE WITH THE"
1820 PRINT "    INGREDIENTS YOU HAVE?"::::::::
1830 CALL SOUND(150,1397,2)
1840 CALL KEY(0,K,S)
1850 IF (K<49)+(K>50)THEN 1840
1860 ON K-48 GOTO 1870,3030
1870 CALL CLEAR
1880 PRINT "DRINK:"::
1890 PRINT :"  (1) MARTINI"
1900 PRINT :"  (2) DRY MARTINI"
1910 PRINT :"  (3) EXTRA DRY MARTINI"
1920 PRINT :"  (4) VODKA MARTINI"
1930 PRINT :"  (5) MANHATTAN"
1940 PRINT :"  (6) DRY MANHATTAN"
1950 PRINT :"  (7) SWEET MANHATTAN"
1960 PRINT :"  (8) PERFECT MANHATTAN"
1970 PRINT :"  (9) WHISKEY SOUR"
1980 PRINT :"  (C) CONTINUE"
1990 CALL SOUND(150,1397,2)
2000 CALL KEY(0,K,S)
2010 IF K=67 THEN 2050
2020 IF (K<49)+(K>57)THEN 2000
2030 II=K-48
2040 GOTO 2230
2050 CALL CLEAR
2060 PRINT "DRINK:"
2070 PRINT :"  (0) WARD EIGHT"
2080 PRINT :"  (1) DAIQUIRI"
```



Continued on p. 86

## Counting Lesson

```
2020 CALL SOUND(100,523,0)
2030 CALL SOUND(100,523,0)
2040 CALL SOUND(100,523,0)
2050 CALL SOUND(100,440,0)
2060 CALL SOUND(100,[illegible]
2070 CALL SOUND(100,[illegible]
2080 CALL SOUND(100,3[illegible]
2090 CALL SOUND(100,440,0)
2100 CALL SOUND(100,349,0)
2110 CALL SOUND(100,262,0)
2120 GOTO 2690
2130 REM  START SHARK FIN ACROSS
     SCREEN IN BACKGROUND
2140 F$="00000103070F1F3F"
2150 CALL HCHAR(22,1,140,32)
2160 CALL CHAR(100,F$)
2170 CALL HCHAR(10,C,100)
2180 CALL SOUND(25,110,5)
2190 CALL HCHAR(10,C,136)
2200 IF C=1 THEN 1760
2210 C=C-1
2220 FB=1
2230 CALL KEY(0,K,STATUS)
2240 IF STATUS=0 THEN 2170
2250 IF K=1 THEN 3080
2260 IF K=15 THEN 160
2270 IF K<48 THEN 2920
2280 IF K>57 THEN 2920
2290 NN=K-48
2300 IF NN<>N THEN 2330
2310 GOTO 1980
2320 REM   SHARK EATS GULL ON
     WRONG ANSWER
2330 R=10
2340 IF SP=1 THEN 2360
2350 CALL SAY("UHOH")
2360 TRIAL=TRIAL+1
2370 F$="FF7E3C1800000000"
2380 CALL CHAR(100,F$)
2390 FOR I=9+(2*N)TO 10 STEP -2
2400 FOR J=1 TO 2
2410 B$="7F3F3F3F1F0F0703"
2420 CALL CHAR(97,B$)
2430 CALL HCHAR(11+P(NM),I,96)
2440 CALL HCHAR(11+P(NM),I+1,98)
2450 CALL HCHAR(12+P(NM),I,97)
2460 CALL HCHAR(12+P(NM),I+1,99)
2470 CALL HCHAR(13+P(NM),I+3,100)
2480 CALL SOUND(50,-5,0)
2490 B$="7F3F1F0F070FFFFF"
2500 CALL CHAR(97,B$)
2510 NEXT J
2520 NM=NM-1
2530 N=N-1
2540 REM  AFTER EATING GULL/WAITS FOR
     CORRECT ANSWER FOR NUMBER LEFT
2550 IF SP=1 THEN 2570
2560 CALL SAY("WHAT+NUMBER+IS+LEFT")
2570 CALL KEY(0,K,S)
2580 IF [illegible] 2970
2590 FOR [illegible] 24
2600 CALL [illegible]LOOP,I,140,32-I)
2610 NEXT [illegible]
2620 IF K=ASC(STR$(N))THEN 2690
2630 TRIAL=TRIAL+1
2640 IF SP=1 THEN 2660
2650 CALL SAY("#THAT IS INCORRECT# A1+A1+A1")
2660 NEXT I
2670 NM=0
2680 IF N<1 THEN 1470
2690 IF SP=1 THEN 2720
2700 CALL SAY("RIGHT")
2710 CALL SAY(CHR$(K))
2720 RIGHT=RIGHT+1
2730 TRIAL=TRIAL+1
2740 NM=0
2750 REM   MOVES SHIP ACROSS SCREEN ONE COLUMN
2760 X=X+1
2770 IF X=17 THEN 2970
2780 CALL HCHAR(9,11,145,21)
2790 CALL HCHAR(8,11,145,21)
2800 CALL HCHAR(9,27-X,120)
2810 CALL HCHAR(9,28-X,121,3)
2820 CALL HCHAR(9,31-X,122)
2830 CALL HCHAR(8,29-X,128)
2840 FOR I=1 TO N
2850 CALL HCHAR(7,29-X,144)
2860 CALL SOUND(200,-5,0)
2870 FOR D=1 TO 50
2880 NEXT D
2890 CALL HCHAR(7,29-X,145)
2900 NEXT I
2910 GOTO 1470
2920 IF SP=1 THEN 2940
2930 CALL SAY("PRESS+A+NUMBER+PLEASE")
2940 TRIAL=TRIAL+1
2950 IF FB=1 THEN 2170 ELSE 1840
2960 REM  ROUTINE WHEN BOAT REACHES ISLAND
2970 X=0
2980 IF SP=1 THEN 3000
2990 CALL SAY("#YOU WIN#")
3000 CALL SOUND(2000,220,0)
3010 CALL SOUND(50,220,30)
3020 CALL SOUND(1000,220,0)
3030 CALL SOUND(50,220,30)
3040 CALL SOUND(500,220,0)
3050 CALL CLEAR
3060 GOTO 1320
3070 REM   CALCULATE SCORES
3080 CALL CLEAR
3090 NM=0
3100 PRINT "TRIALS=";TRIAL::"RIGHT=";RIGHT::"PERCENT CORRECT=";
     INT(100*(RIGHT/TRIAL))
3110 PRINT :"TO RETURN TO LESSON PRESS 1":"TO END LESSON PRESS 2"::
3120 CALL KEY(0,K,S)
3130 IF S=0 THEN 3120
3140 IF K=49 THEN 850
3150 IF K<>50 THEN 3120
3160 PRINT "SO LONG FOR NOW!"
3170 END
```



## Business . . . from p. 74

On the other hand, a company where I helped install order and inventory processing listened very carefully to what we told them about operation for accuracy. They were not only willing to tighten some of their operations, but were also eager to be able to control their warehouse functions more closely. That company is still prospering.

There's another side to consider too. If you have a small list of products to keep track of, you probably do rather well keeping track of them already. And as for calculating optimal inventory levels, reorder quantities, etc., you can probably do that rather well with a pocket calculator and formulas you can find in many textbooks. So honestly, *would* the computer help you do a better job of managing inventory than you already do (or could do manually with the same effort you would have to put into a computer system)? If the answer is no, then save yourself money, time, and management energy by *not* doing computer inventory. If there are real benefits you would receive, I hope this column may make you a little more aware of what you must prepare for, and strengthen your resolve to do it carefully, and do it accurately. The stakes are too high for you to wander casually into a computer inventory system!

Secret of PRK . . . from p. 72

## 3. ACCEPT AT – numerical data

The ACCEPT AT statement works similar to INPUT but can be formatted anywhere on the screen. The input prompt can be printed in the appropriate place with the technique of section 2, above. The built-in value checks are an additional feature:

CALL A(R,C,L,F,A,MN,MX)

R, C and L have been explained in section 1.

| | |
|---|---|
| F = | function variable |
| A = | accept variable |
| MN = | minimum value |
| MX = | maximum value |

F The numerical variable in this position assumes a value 1-7 depending on certain function keys being depressed. The values connected to these functions in this way should not be confused with the ASCII values of these functions that can be useful in CALL KEY statements. For completeness, I'll also tabulate the ASCII values here.

| Function Key | | CALL A value (F position) | ASCII value |
|---|---|---|---|
| TI-99/4A | TI-99/4 | | |
| FCTN 5 | SHIFT W – BEGIN | 6 | 14 |
| FCTN 8 | SHIFT R – REDO | 4 | 6 |
| FCTN 7 | SHIFT A – AID | 3 | 1 |
| FCTN 9 | SHIFT Z – BACK | 7 | 15 |
| FCTN 4 | SHIFT C – CLEAR | 2 | 2 |
| FCTN 6 | SHIFT V – PROC'D | 5 | 12 |
| | ENTER | 1 | 13 |

CLEAR will not only give F a value of 2, but it also clears the input printing field on the screen, and is to be used when typed input is not yet entered and should be changed. *Warning:* This means that if you write a program that continually loops to a CALL A statement, CLEAR cannot be used to break the program. Only QUIT or cutting the power will work then, but it will also erase your program in the process! The solution to this problem is to program your escape routine—e.g., IF F=3 THEN 10000 enabling you to use AID to bring the program to line 10000 which reads : 10000 END.

A The variable in the position of A assumes (accepts) the value you typed in much in the same way as the input variable does after you depress ENTER. The F variable, of course, then gets the value 1 since you have used the function key ENTER. If pressing ENTER when the print/input field contains no information (only "space"), F will take on the value in the above table, if one of the function keys has previously been pushed.

X The numbers or the values of the numerical variables in the positions MN and MX respectively determine the minimum and maximum values that A will accept. A gentle beep when pressing ENTER warns you if you try to step beyond these imposed limits. The screen, of course, will accept any numerical data, provided that the length does not exceed L (e.g., if L=2 and MX=10000 you still cannot get A to become more than 99 since the screen will not accept more than 2 digits). Since the plus and minus signs (+ and –) as well as the letter E (scientific notation) are all considered to be *numerical* input, they will also be accepted. String data, however, are not accepted by the screen at all when using CALL A in this way.

If MN=MX A will only accept the MN and MX value.
If MN>MX A shouldn't accept any value at all, but illogically it does accept the MN value.

### Example 3

```
100  CALL  CLEAR
110  CALL  D(3,3,28,"ENTER 1,2 OR 3")
120  CALL  A(10,25,1,F,B,1,3)
130  CALL  CLEAR
140  FOR  T=1  TO  500
150  NEXT T
160  CALL  D(15,3,28,"YOUR CHOICE WAS")
170  CALL  D(15,20,2,B)
180  FOR  T=1  TO  500
190  NEXT  T
200  GOTO  100
```

## 4. ACCEPT AT – string data

CALL A(R,C,L,F,A$)

R, C and L are explained in section 1.
F is explained in section 3.
A$ = accept string variable

A$ The variable in the A$ position is filled with the typed string information when pressing ENTER.

### Example 4

```
100  CALL  CLEAR
110  M$="PLEASE ENTER YOUR NAME"
120  CALL  D(5,3,26,M$)
130  CALL  A(10,3,20,F,N$)
140  CALL CLEAR
150  FOR T=1 TO 500
160  NEXT T
170  CALL  D(5,2,28,"THANKS "&N$)
180  FOR T=1 TO 500
190  NEXT T
200  GOTO 100
```

That should do it, fellow 99'ers. I've told you some "personal" secrets, and shown you their "BASIC" uses. Now you're on your own: It's your turn to apply these two new commands and perhaps, discover some additional ones.

[**Note:** In the event that Texas Instruments gets away from producing "hybrid" Command Modules (containing *both* BASIC and GPL coding), future releases of *Personal Record Keeping* will not offer the capabilities described in this article—Ed.]

Super Language . . . from p. 76

The Utility Routines VMBW, VSBW, VWTR, and VMBR are used in the example program. The VDP Multiple Byte Write (VMBW) moves the number of bytes in register 2 (R2) from the CPU RAM address in R1 to the VDP RAM address in R0. VSBW, the VDP Single Byte Write routine, was explained earlier. VDP Write To Register (VWTR) puts the value that is in the rightmost byte of R1 into the VDP register whose number is in the leftmost byte of R1. Among other things, these VDP registers are used to select VDP modes and features. VMBR is the VDP Multiple Byte Read routine, which reads the number of bytes specified in R2 into the CPU RAM location in R1 from the VDP RAM location in R0.

The logic for detecting hits in the Assembly Language program is based on the fact that the point of the arrow is three pixels to the right and seven pixels below the corner of the sprite which is obtained from the Sprite Attribute Block.

### Conclusion

Although they are more complex to write, Assembly Language programs are far superior to BASIC programs when it comes to execution speed and for controlling the facilities of the 99/4A computer. In some cases, as in the game of *Life*, the faster speed of Assembly Language turns a boring game into one which is fun to watch. In other cases, as in the program SHOOT, Assembly Language is capable of providing more accurate results. Thus, having the capability to write programs or subroutines in Assembly Language lets you achieve results which are impossible with BASIC and Extended BASIC alone.

[**Note:** See discussion of the Editor/Assembler in "TI at the Consumer Electronics Show" in this issue—Ed.]

## Bartender . . . from p. 84

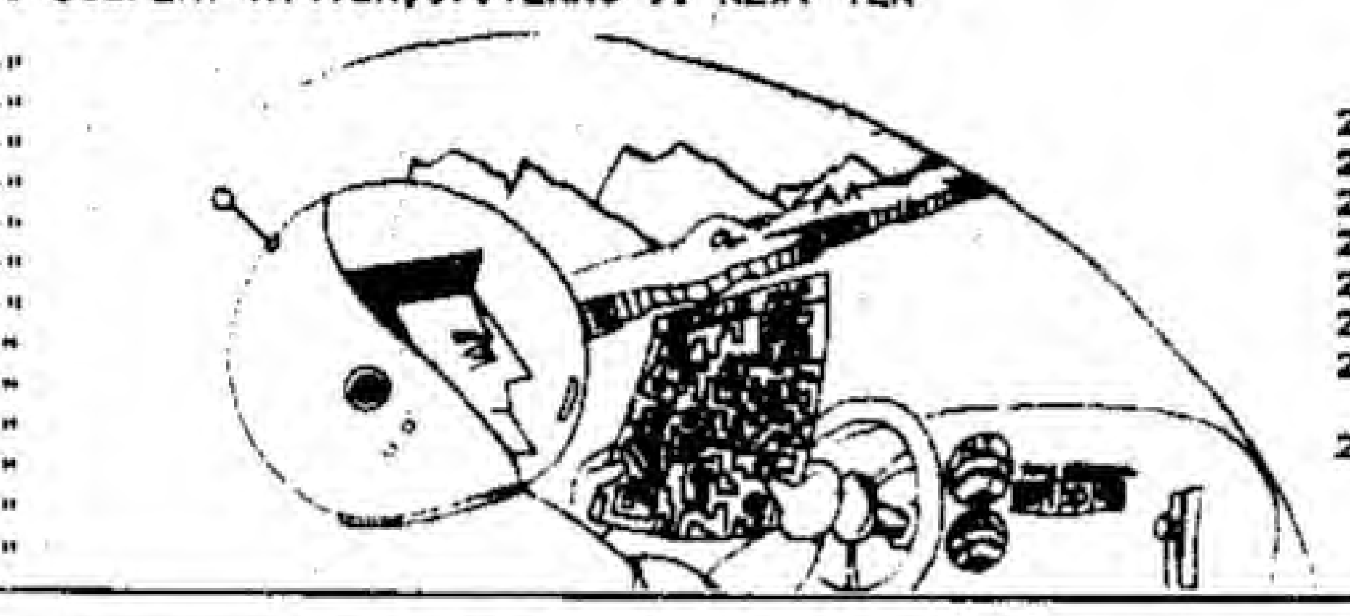
```
2090 PRINT :"  (2) BACARDI"
2100 PRINT :"  (3) SCREWDRIVER"
2110 PRINT :"  (4) PINK LADY"
2120 PRINT :"  (5) SALTY DOG"
2130 PRINT :"  (6) GIN COOLER"
2140 PRINT :"  (7) TOM COLLINS"
2150 PRINT :"  (8) BLACK RUSSIAN"
2160 PRINT :"  (9) OLD-FASHIONED"
2170 PRINT :"  (R) REPEAT FIRST SCREEN"
2180 CALL SOUND(150,1397,2)
2190 CALL KEY(0,K,S)
2200 IF K=82 THEN 1870
2210 IF (K<48)+(K>57)THEN 2190
2220 II=K-38
2230 CALL CLEAR
2240 C=VAL(DRINK$(II,1))
2250 GOSUB 250
2260 PRINT "** ";DRINK$(II,0)::::
2270 ON VAL(DRINK$(II,2))GOTO 2280,2310,
     2340,2370,2420,2450
2280 PRINT "FILL MIXING GLASS"
2290 PRINT "2/3 FULL WITH ICE CUBES"
2300 GOTO 2550
2310 PRINT "FILL HIGHBALL GLASS"
2320 PRINT "FULL WITH ICE CUBES"
2330 GOTO 2550
2340 PRINT "SALT RIM OF HIGHBALL GLASS"
2350 PRINT "FILL WITH ICE"
2360 GOTO 2550
2370 PRINT "FILL TALL FROSTED"
2380 PRINT "GLASS WITH ICE"
2390 PRINT "SQUEEZE 1/4 LIME"
2400 PRINT "OVER ICE-DROP IN"
2410 GOTO 2550
2420 PRINT "FILL OLD-FASHIONED GLASS"
2430 PRINT "WITH ICE"
2440 GOTO 2550
2450 PRINT "1 CUBE OF SUGAR IN"
2460 PRINT "OLD-FASHIONED GLASS"
2470 PRINT "2-3 DROPS ANGOSTURA BITTERS"
2480 PRINT "OVER CUBE OF SUGAR"
2490 PRINT "RIM GLASS WITH LEMON TWIST"
2500 PRINT "DROP IN":"1/2 OZ. SODA OR WATER"
2510 PRINT "MUDDLE THOROUGHLY"
2520 PRINT "FILL WITH ICE"
2530 PRINT "1 OZ. BOURBON":"STIR"
2540 GOTO 2720
2550 FOR JJ=8 TO 23
2560 IF DRINK$(II,JJ)="" THEN 2580
2570 PRINT DRINK$(II,JJ);INV$(JJ-8,0)
2580 NEXT JJ
2590 ON VAL(DRINK$(II,3))GOTO 2720,2600,
     2620,2640
2600 PRINT "STIR AND STRAIN INTO"
2610 GOTO 2660
2620 PRINT "SHAKE AND STRAIN INTO"
2630 GOTO 2660
2640 PRINT "STIR LIGHTLY"
2650 GOTO 2720
2660 ON VAL(DRINK$(II,4))GOTO 2720,2670,
     2690,2710
2670 PRINT "3 OZ. CHILLED COCKTAIL GLASS"
2680 GOTO 2720
2690 PRINT "5 OZ. CHILLED COCKTAIL GLASS"
2700 GOTO 2720
2710 PRINT "WHISKEY SOUR GLASS"
2720 ON VAL(DRINK$(II,5))GOTO 2800,2730,
     2760,2790
2730 PRINT "GARNISH WITH SPIKED OLIVE"
2740 CALL COLOR(15,13,C)
2750 GOTO 2800
2760 PRINT "GARNISH WITH SPIKED CHERRY"
2770 CALL COLOR(15,10,C)
2780 GOTO 2800
2790 PRINT "GARNISH WITH LEMON TWIST"
2800 IF DRINK$(II,6)="1" THEN 2820
2810 PRINT "AND ORANGE SLICE"
2820 ON VAL(DRINK$(II,7))GOTO 2860,2830,
     2850
2830 PRINT "SERVE WITH STIR ROD"
2840 GOTO 2860
2850 PRINT "SERVE WITH TWO STRAWS"
2860 IF DRINK$(II,2)="1" THEN 2890
2870 ON VAL(DRINK$(II,2))-1 GOSUB 310,310,
     700,850,850
2880 GOTO 2900
2890 ON VAL(DRINK$(II,4))-1 GOSUB 460,460,
     1000
2900 ON VAL(DRINK$(II,5))GOSUB 290,1210,
     1210,1250
2910 IF DRINK$(II,6)="1" THEN 2930
2920 GOSUB 1290
2930 ON VAL(DRINK$(II,7))GOTO 2980,2940,
     2970
2940 CALL VCHAR(3,26,96,4)
2950 CALL HCHAR(4,26,110)
2960 GOTO 2980
2970 CALL VCHAR(3,26,99,4)
2980 IF DRINK$(II,2)<>"3" THEN 3010
2990 CALL COLOR(16,16,8)
3000 CALL HCHAR(4,23,158,5)
3010 CALL KEY(0,K,S)
3020 IF S=0 THEN 3010 ELSE 1760
3030 CALL CLEAR
3040 PRINT "IN THE FOLLOWING LIST,"
3050 PRINT "PRESS ""Y"" IF YOU HAVE"
3060 PRINT "THE INGREDIENT."
3070 PRINT "PRESS ""N"" IF YOU DO NOT."
3080 PRINT :"PRESS ""B"" TO BACK UP"::::::
3090 CALL SOUND(150,1397,2)
3100 YS=0
3110 FOR KK=0 TO 15
3120 PRINT "_ ";INV$(KK,0)
3130 CALL KEY(0,KEY,S)
3140 IF KEY=66 THEN 3030
3150 IF KEY=78 THEN 3180
3160 IF KEY<>89 THEN 3130
3170 YS=YS+1
3180 CALL HCHAR(23,3,KEY)
3190 INV$(KK,1)=CHR$(KEY)
3200 NEXT KK
3210 DR=0
3220 PRINT ::"YOU CAN MAKE:"::
3230 IF YS>1 THEN 3270
3240 PRINT "NOTHING; SORRY."::"YOU NEED
     TO GO TO THE LIQUOR"
3250 PRINT "STORE IF YOU'RE THIRSTY."
3260 GOTO 3380
3270 FOR I=1 TO 19
3280 FOR J=8 TO 23
3290 IF DRINK$(I,J)="" THEN 3310
3300 IF INV$(J-8,1)="N" THEN 3350
3310 NEXT J
3320 PRINT DRINK$(I,0)
3330 CALL SOUND(150,1397,2)
3340 DR=DR+1
3350 NEXT I
3360 IF DR=0 THEN 3240 .
3370 PRINT :"THAT'S ALL."
3380 PRINT :"PRESS ANY KEY TO CONTINUE"
3390 CALL KEY(0,K,S)
3400 IF S=0 THEN 3390 ELSE 1760
3410 DATA MARTINI,16,1,2,2,2,1,1
3420 DATA "1 OZ. ",,,"1/3 OZ.",,,,,,,,,,,,""
3430 DATA "DRY MARTINI",16,1,2,2,2,1,1
3440 DATA "1 1/4 OZ. ",,,"1/4 OZ. ",,,,,,,,,,,,""
3450 DATA "EXTRA DRY MARTINI",16,1,2,2,2,1,1
3460 DATA "1 1/2 OZ. ",,,"2-3 DROPS ",,,,,,,,,,,,""
3470 DATA "VODKA MARTINI",16,1,2,2,2,1,1
3480 DATA "","1 OZ. ",,"1/3 OZ.",,,,,,,,,,,,""
3490 DATA MANHATTAN,7,1,2,2,3,1,1
3500 DATA "",,"1 OZ. ",,,"1/2 OZ. ",,,,"2-3 DROPS ",,,,,,""
3510 DATA "DRY MANHATTAN",7,1,2,2,2,1,1
3520 DATA "",,"1 OZ. ","1/2 OZ. ",,,,,,,,,,,,""
3530 DATA "SWEET MANHATTAN",7,1,2,2,3,1,1
3540 DATA "",,"1 OZ. ",,,"1/2 OZ. ",,,"1/4 OZ. ",
     "2-3 DROPS ",,,,,,""
3550 DATA "PERFECT MANHATTAN",7,1,2,2,4,1,1
3560 DATA "",,"1 OZ. ","1/4 OZ. ",,"1/4 OZ. ",,,,
     "2-3 DROPS ",,,,,,""
3570 DATA "WHISKEY SOUR",12,1,3,4,3,2,1
3580 DATA "",,"1 OZ. ",,,,,"1 OZ. ","1/2 OZ. ",,,,,,,""
3590 DATA "WARD EIGHT",7,1,3,4,3,2,1
3600 DATA "",,"1 OZ. ",,,,,"1 OZ. ",,,"1/2 OZ. ",,,,,""
3610 DATA "DAIQUIRI",16,1,3,3,1,1,1
3620 DATA "",,,,"1 OZ. ",,,"1 OZ. ","1/2 OZ. ",,,,,,,""
3630 DATA BACARDI,7,1,3,3,1,1,1
3640 DATA "",,,,"1 OZ. ",,,"1 OZ. ",,,"1/2 OZ. ",,,,,""
3650 DATA SCREWDRIVER,11,2,1,1,1,1,2
3660 DATA "","1 OZ. ",,,,,,,,,,,"FILL WITH ",,,""
3670 DATA "PINK LADY",10,1,3,3,1,1,3
3680 DATA "1 OZ. ",,,,,,,,,,"1/2 OZ. ",,,,,"1 1/2 OZ. "
3690 DATA "SALTY DOG",16,3,4,1,1,1,2
3700 DATA "1 OZ. ",,,,,,,,,,,,"FILL WITH ",,,,""
3710 DATA "GIN COOLER",7,4,4,1,3,2,3
3720 DATA "1 OZ. ",,,,,,,,"1 OZ. ",,,"1/2 OZ. ",,,
     "FILL WITH ",,""
3730 DATA "TOM COLLINS",16,4,4,1,3,2,3
3740 DATA "1 OZ. ",,,,,,,,"1 OZ. ","1/2 OZ. ",,,,,,
     "FILL WITH ",""
3750 DATA "BLACK RUSSIAN",2,5,4,1,1,1,1
3760 DATA "","1 OZ. ",,,,,,"1/2 OZ. ",,,,,,,,,""
3770 DATA "OLD-FASHIONED",7,6,1,1,3,2,1
3780 DATA "",,"1 OZ. ",,,,,,,,"2-3 DROPS ",,,,,"1/2 OZ. ",""
3790 DATA GIN,VODKA,BOURBON,"DRY VERMOUTH","LIGHT RUM",
     "SWEET VERMOUTH",KAHLUA
3800 DATA "LEMON JUICE","SIMPLE SYRUP","ANGOSTURA BITTERS",
     GRENADINE,"GRAPEFRUIT JUICE"
3810 DATA "ORANGE JUICE","GINGER ALE","SODA","MILK OR CREAM"
```

## Mystery Words . . . from p. 61

```
1070 MM=275
1080 GOSUB 4710
1090 GOSUB 4840
1100 CALL SOUND(1200,262,1,311,1,392,1)
1110 CALL CLEAR
1120 FOR I=1 TO 500
1130 NEXT I
1140 FOR I=1 TO 99
1150 READ D$
1160 CALL HCHAR(VAL(SEG$(D$,1,2)),VAL(SEG$(D$,3,2)),40)
1170 NEXT I
1180 FOR I=100 TO 188
1190 READ D$
1200 CALL HCHAR(VAL(SEG$(D$,1,2)),VAL(SEG$(D$,3,2)),35)
1210 NEXT I
1220 DATA 0503,0603,0703,0803,0903,1003,1103,0604,0705,0606,0507,
     0607,0707,0807,0907,1007
1230 DATA 1107,0107,0207,0308,0111,0211,0310,0409,0509,0609,0709,
     0809,0714,0713,0712,0711
1240 DATA 0811,0911,1011,1012,1013,1014,1114,1214,1314,1313,1312,
     1311,0314,0515,0516,0517
1250 DATA 0518,0616,0716,0816,0916,1016,1116,0721,0720,0719,0718,
     0818,0918,1018,1118,1218
1260 DATA 1318,1319,1320,1321,1019,1020,0523,0623,0723,0823,0923,
     1023,1123,0524,0525,0526
1270 DATA 0626,0726,0826,0825,0824,0924,1025,1126,0126,0226,0327,
     0130,0230,0329,0428,0528,0628
1280 DATA 0728,0828,1504,1604,1704,1804,1904,2004,2104,2005,1906,
     2007,2108,2008,1908,1808
1290 DATA 1708,1608,1508,1813,1812,1811,1810,1910,2010,2110,2210,
     2310,2410,2411,2412,2413
1300 DATA 2313,2213,2113,2013,1913,1615,1715,1815,1915,2015,2115,
     2215,1616,1617,1618,1718
1310 DATA 1818,1918,1917,1916,2016,2117,2218,1921,2021,2121,2221,
     2321,1820,1821,1822,1823
1320 DATA 1824,1924,2024,2124,2224,2324,2424,2423,2422,2421,2420,
     1729,1728,1727,1726,1826
1330 DATA 1926,2026,2027,2028,2029,2129,2229,2329,2328,2327,2326
1340 FOR I=1 TO 59
1350 READ SET$(I)
1360 NEXT I
1370 DATA 175,135,144,125,214,211,217,255,257,254,251,312,412,414,
     452,455,522,577,612,614,654,717,755,712,1754
1380 DATA 2125,2514,2556,3165,3175,4514,4516,4554,5475,6135,6145,
     6554,7175,655,31754,456135,453145,3122175
1390 DATA 2177175,21475,14454,412254,54754,217754,257754,52254,
     61354,2145,133545,14175,217414,4531,613145,566135
1400 FOR I=1 TO 2
1410 FOR J=1 TO 7
1420 FOR K=1 TO 2
1430 READ LINE(J,I,K)
1440 NEXT K
1450 NEXT J
1460 NEXT I
1470 DATA 5,0,-5,0,4,0,-4,0,3,-7,-3,6,-6,0,-3,6,-6,0,5,0,-5,0,4,
     0,-4,0,3,-7
1480 CALL CHAR(96,"0406050404040404")
1490 CALL CHAR(40,"")
1500 CALL CHAR(97,"000000804020101")
1510 CALL CHAR(98,"0101010101010101")
1520 CALL CHAR(99,"FF04040404040404")
1530 CALL CHAR(100,"FF080810102040B")
1540 CALL CHAR(101,"FF00000001020408")
1550 CALL CHAR(102,"FF061C6484040404")
1560 CALL CHAR(103,"FF")
1570 CALL CHAR(104,"FF10101010101010")
1580 CALL CHAR(105,"FF0F344484848444")
1590 CALL CHAR(106,"FF00C02010100808")
1600 CALL CHAR(107,"FF080804040201")
1610 CALL CHAR(108,"FF0404040404847F")
1620 CALL CHAR(109,"FF08080810204080")
1630 CALL CHAR(110,"FF00000304080810")
1640 CALL CHAR(111,"FF00FC03")
1650 CALL CHAR(112,"FF0000038340202")
1660 CALL CHAR(113,"FF100804")
1670 CALL CHAR(114,"FF1010101313101")
1680 CALL CHAR(115,"FF00000000000102")
1690 CALL CHAR(116,"FF20204040B")
1700 CALL CHAR(117,"FF00000000010608")
1710 CALL CHAR(118,"FF041820C")
1720 CALL CHAR(120,"FF00000000384482")
1730 CALL CHAR(121,"FF824438")
1740 CALL CHAR(119,"FF38448282824438")
1750 CALL CHAR(122,"0038448282824438")
1760 CALL CHAR(123,"0000000000384482")
1770 CALL CHAR(136,"")
1780 CALL CHAR(137,"3C7EFFFFFFFF7E3C")
1790 CALL CHAR(138,"FFFFFFFFFFFFFFFF")
1800 CALL CHAR(128,"3C7EFFFFFFFF7E3C")
1810 CALL CHAR(95,"")
1820 FOR I=5 TO 12
1830 CALL COLOR(I,2,16)
1840 NEXT I
1850 IF FLAG=0 THEN 360 ELSE 1860
1860 CALL COLOR(13,5,5)
1870 CALL COLOR(2,7,7)
1880 CALL COLOR(3,2,16)
1890 CALL COLOR(4,2,16)
1900 CALL COLOR(13,15,1)
1910 CALL COLOR(14,15,1)
1920 GOSUB 3830
1930 C=13
1940 TRY=1
1950 FOR I=5 TO 12
1960 CALL COLOR(I,16,16)
1970 NEXT I
1980 CALL HCHAR(2,13,95,13)
1990 FOR I=3 TO 7
2000 CALL HCHAR(I,13,103,13)
2010 NEXT I
2020 CALL HCHAR(9,13,95,13)
2030 IF CLEF<>3 THEN 2080
2040 CALL HCHAR(10,13,95,13)
2050 FOR I=11 TO 15
2060 CALL HCHAR(I,13,103,13)
2070 NEXT I
2080 CALL HCHAR(20,3,95,12)
2090 CALL HCHAR(20,19,95,12)
2100 N=INT(NSET*RND)+1
2110 WORD$=SET$(N)
2120 FOR I=1 TO LEN(WORD$)
2130 N=1
2140 FOR K=1 TO 3
2150 NLINE(K)=0
2160 NEXT K
2170 IF CLEF=2 THEN 2250
2180 FOR J=1 TO 2
2190 L=LINE(VAL(SEG$(WORD$,I,1)),1,J)
2200 IF L=0 THEN 2230
2210 NLINE(N)=L
2220 N=N+1
2230 NEXT J
2240 IF CLEF=1 THEN 2330
2250 FOR J=1 TO 2
2260 L=LINE(VAL(SEG$(WORD$,I,1)),2,J)
2270 IF L=0 THEN 2320
2280 IF CLEF=2 THEN 2300
2290 L=SGN(L)*(ABS(L)+8)
2300 NLINE(N)=L
2310 N=N+1
2320 NEXT J
2330 N=N-1
2340 IF N=1 THEN 2360
2350 N=INT(N*RND)+1
2360 L=NLINE(N)
2370 IF L<0 THEN 2450
2380 IF L=2 THEN 2420
2390 IF L=10 THEN 2420
2400 CALL HCHAR(L,C,119)
2410 GOTO 2430
2420 CALL HCHAR(L,C,122)
2430 C=C+2
2440 GOTO 2530
2450 L=ABS(L)
2460 IF L=3 THEN 2500
2470 IF L=11 THEN 2500
2480 CALL HCHAR(L-1,C,120)
2490 GOTO 2510
2500 CALL HCHAR(L-1,C,123)
2510 CALL HCHAR(L,C,121)
2520 C=C+2
2530 NEXT I
2540 ROW=20
2550 MSG$="READY"
2560 COL=3
2570 GOSUB 4600
2580 COL=19
2590 GOSUB 4600
2600 CALL COLOR(13,15,1)
2610 CALL HCHAR(12,3,137)
2620 CALL HCHAR(12,30,137)
2630 CALL HCHAR(8,3,128)
2640 CALL HCHAR(8,30,128)
2650 CALL COLOR(13,7,1)
2660 CALL SOUND(100,880,1)
2670 FOR I=5 TO 8
2680 CALL COLOR(I,2,16)
2690 NEXT I
2700 CALL KEY(1,KEY1,S)
2710 CALL KEY(2,KEY2,S)
2720 IF KEY1=19 THEN 2750
2730 IF KEY2=10 THEN 2850
2740 GOTO 2700
2750 CALL COLOR(13,15,1)
2760 CALL HCHAR(8,3,137)
2770 CALL HCHAR(8,30,137)
2780 CALL HCHAR(10,3,128)
2790 CALL HCHAR(10,30,128)
```

## InterplanetaryRescue . . . from p. 61

```
1240 IF ANS$="N" THEN 1260 ELSE IF ANS$<>"Y" THEN 1230
1250 CALL CLEAR :: GOTO 170
1260 STOP
1270 CALL HCHAR(22,1,32,96)
1280 IF XC<>137 AND YC<>185 THEN GOSUB 1160 :: DISPLAY AT(22,1):"YOU MISSED THE PAD.":"YOUR SHIP HAS CRASHED."
1290 IF V>-30 THEN 1330 ELSE GOSUB 1160 :: DISPLAY AT(22,1):"YOU BLEW IT. YOU LEFT A"
1300 DISPLAY AT(23,1):"CRATER A MILE WIDE."
1310 DISPLAY AT(24,1):"VELOCITY=";V
1320 GOTO 1190
1330 IF V>-20 THEN 1370 ELSE GOSUB 1160 :: DISPLAY AT(22,1):"A BAD LANDING-TWO CREWMEN"
1340 DISPLAY AT(23,1):"ARE DEAD, AND YOU ARE HURT."
1350 DISPLAY AT(24,1):"VELOCITY=";V
1360 GOTO 1190
1370 IF V>-10 THEN 1410 ELSE GOSUB 1160 :: DISPLAY AT(22,1):"YOUR SHIP IS BADLY DAMAGED."
1380 DISPLAY AT(23,1):"THIS IS YOUR LAST FLIGHT."
1390 DISPLAY AT(24,1):"VELOCITY=";V
1400 FF=FF-3000 :: GOTO 1190
1410 IF V>-6 THEN 1430 ELSE CALL CLEAR :: CALL CHARSET :: DISPLAY AT(22,1):"A ROUGH LANDING. YOU HAVE"
1420 DISPLAY AT(23,1):"LOST 1/2 OF YOUR FUEL." :: E=E/2 :: FF=2000 :: GOTO 1450
1430 CALL CLEAR :: CALL CHARSET :: DISPLAY AT(22,1):"A PERFECT LANDING. YOU ARE"
1440 DISPLAY AT(23,1):"IN GOOD SHAPE TO RETURN." :: FF=3000
1450 DISPLAY AT(24,1):"VELOCITY=";V :: FOR TD=1 TO 500
1460 NEXT TD :: F=0 :: TRIP=2 :: GOSUB 190 :: GOSUB 1660 :: GOTO 370
1470 CALL DELSPRITE(#1,#2,#3)
1480 CALL CLEAR
1490 CALL CHARSET :: IF V<=-10 THEN 1270 ELSE CALL HCHAR(22,1,32,96)
1500 IF V>-6 THEN 1530 ELSE IF D3<>25 OR D4<>41 THEN 1270
1510 DISPLAY AT(22,1):"A ROUGH LANDING. YOUR SHIP  BARELY MADE IT."
1520 FF=FF+4000 :: GOTO 1550
1530 DISPLAY AT(22,1):"CONGRATULATIONS, A PERFECT"
1540 DISPLAY AT(23,1):"LANDING. EARTH IS PROUD."
1550 DISPLAY AT(24,1):"VELOCITY=";V :: FOR TD=1 TO 2000
1560 NEXT TD :: GOTO 1190
1570 CALL CLEAR :: GOSUB 2360 :: CALL CLEAR :: DISPLAY AT(1,7):"PLANET OPTIONS"
1580 DISPLAY AT(3,1): :"1. MOON": :"2. MARS": :"3. VENUS"
1590 ACCEPT AT(10,1)VALIDATE("123")SIZE(1):OPT1 :: CALL CLEAR
1600 DISPLAY AT(1,4):"LEVEL OF DIFFICULTY"
1610 DISPLAY AT(3,1): :"1. BEGINNER": :"2. INTERMEDIATE": :"3. NOVICE": :"4. PROFESSIONAL"
1620 ACCEPT AT(12,1)VALIDATE("1234")SIZE(1):OPT2
1630 IF OPT1=1 THEN G=2 :: E=20000 :: TOFF=65000 :: GOTO 1660
1640 IF OPT1=2 THEN G=4 :: E=45000 :: TOFF=230000 :: GOTO 1660
1650 IF OPT1=3 THEN G=6 :: E=80000 :: TOFF=540000
1660 IF OPT2=1 THEN RESTORE 1730 :: GOTO 1700
1670 IF OPT2=2 THEN RESTORE 1940 :: GOTO 1700
1680 IF OPT2=3 THEN RESTORE 470 :: GOTO 1700
1690 IF OPT2=4 THEN RESTORE 2150
1700 CALL CLEAR :: CALL COLOR(8,10,12)
1710 FOR TER=1 TO 21 :: READ TERN$ :: DISPLAY AT(TER,1):TERN$ :: NEXT TER
1720 GOSUB 690 :: RETURN
1730 DATA "!!^^!!!!^^^^^^^^^^^_^^^^^^"
1740 DATA "!!!!!!!!!^^^^^^^^_^_^^^^^"
1750 DATA "!!!T!!!!!^^^^^^^___^^^"
1760 DATA "!!!>!!!!!!^^^^^^^______^^"
1770 DATA "!!!!!!!!!!!!!!^^^^__$$__^^^"
1780 DATA "!!!!!!!!!!!!!!^^^______^^^^"
1790 DATA "!!!!!!!!!!!!^^^_^^_____^^^^"
1800 DATA "!!!!!!!!!!!!^^^__^___^^^^^^^"
1810 DATA "^!!!!!!!!!^^^^^_^^^^^^^^^^^"
1820 DATA "^^^!!!^^^^^^^^^^^^__^^^^!^^"
1830 DATA "^^^!^^^^^^^^^_^^____^^!!^^^"
1840 DATA "^^^^^^^^^_^__________^!!!!^"
1850 DATA "^^^^^^^^__________^^^^!!!!!"
1860 DATA "^^^^^^^____$$___^^^^!!!!!"
1870 DATA "^^^^^^^^__________^^^^^!!!!!"
1880 DATA "^^_^^^^^__^___^^^^^^!!!!!"
1890 DATA "____^__^^_^^_^^^^^^!!!!!!"
1900 DATA "__________^^^^^^^^^^^^!!!>!!!"
1910 DATA "_$_______^^___^^^^^^^!!!!!!!!"
1920 DATA "$$$$___________^^^^!!!!!!!!!"
1930 DATA "$77$$_________^^^^^!!!!!!!!"
1940 DATA "!!!!!!!!!!!!!!!!!!^^^^^^^^^^^^"
1950 DATA "!!!!!!!!!^^!!!^^^^^___^^^^^"
1960 DATA "!!!!!!!!^^!!!!!^^^^____^^^^^"
1970 DATA "!!!>!^^^^!!!^^^^^^^____^^^^^"
1980 DATA "^^!!!^^^^!!^^^^^______^^____"
1990 DATA "^^!!!^__^^!^^^^____$________"
2000 DATA "^^^^^^^_^^^!!^^^___$$______^^"
2010 DATA "^^^^^^^^^^^!!!^^_____$____^^^"
2020 DATA "^^^^^__^^!!^^___________^^^^^^^"
2030 DATA "_^^_____^^^^______^_^^^^^^^^^"
2040 DATA "$___$$_^^^_____^^^^^!!!^^^^"
2050 DATA "$_^___$$___^^_____^!!^!!!^^^^^"
2060 DATA "_^^^___$____^_____^^!!!!^^^^^"
2070 DATA "^^^^^^_____^^$$___^!!!!^^^^^^"
2080 DATA "^^^^^^X____^$_____^!^!^^^^^^^"
2090 DATA "^^^^^^_____^$___^^!!!!!!!^^^"
2100 DATA "^^^^^^______$$___^!!!^^!!!!!^"
2110 DATA "^^^^^^^^_______^^^!^^^!!>!!^"
2120 DATA "^^^^^^^^^^^^^^^^^^^^^^^^!!!^^^"
2130 DATA "^^^^^^^^^^^!^^^^^^^^^^^^^^^^^"
2140 DATA "^^^^^^^^^^!!!^^^^^^^^^^^^^^^^"
2150 DATA "_________________$$$$$$77$$"
2160 DATA "$^^^^^___^^____^___$$$$7777$"
2170 DATA "$^!!!^____^___^^____$$77777$"
2180 DATA "_^!>!^____^_^^^_____$77777$"
2190 DATA "_^!!!^^^^^^^^^^^__$$$$$7777$"
2200 DATA "_^^^^^____^____$$$777$$77$$"
2210 DATA "_$$_____$$_____$$$7777$$$$$"
2220 DATA "$$$$$$$$$$_$$_$$777777$$77"
2230 DATA "$$77$$_____$$_$77777777777"
2240 DATA "$7777$_$$$$$_$77777777$$77"
2250 DATA "77$$$$_$77777777777777$777"
2260 DATA "77$_____$$$7777777777$$77$"
2270 DATA "$$$_$$$$7777$$777777$$7$$"
2280 DATA "$$__$77777777777$$$$$$$$$"
2290 DATA "$$_$77$$$$7777$$$$_$$$___"
2300 DATA "$$___7$77$7777$$___$^^^^_"
2310 DATA "$$$$_$$7$$7777____^_^!!!^_"
2320 DATA "$$$$7777$7[illegible]77_77^^^!>!^_"
2330 DATA "$$$77777[illegible]777_^!!!^_"
2340 DATA "$777777777$[illegible]"
2350 DATA "$$$7777777[illegible]"
```



```
2360 DISPLAY AT(11,2):"$$$$$$$$$$$$$$$$$$$[illegible]"
2370 DISPLAY AT(12,2):"$ INTERPLANETARY [illegible]"
2380 DISPLAY AT(13,2):"$$$$$$$$$$$$$$$$$$$[illegible]"
2390 CALL SOUND(-4250,110,7,220,7,110,7,-8,7)
2400 DISPLAY AT(24,1):"PRESS ANY KEY TO [illegible]"
2410 CALL SPRITE(#2,98,2,180,125,-3,0,#3,100,7,[illegible],128,-3,0)
2420 CALL KEY(0,KEY,STAT):: IF STAT=0 THEN CALL [illegible](-1000,
     110,7,220,7,110,7,-5,7):: GOTO 2420
2430 CALL DELSPRITE(#2,#3):: RETURN
```

## Mystery Words

```
2800 CALL COLOR(13,11,1)
2810 CALL SOUND(100,880,1)
2820 CALL KEY(2,KEY2,S)
2830 IF KEY2<>10 THEN 2820
2840 GOTO 2940
2850 CALL COLOR(13,15,1)
2860 CALL HCHAR(8,3,137)
2870 CALL HCHAR(8,30,137)
2880 CALL HCHAR(10,3,128)
2890 CALL HCHAR(10,30,128)
2900 CALL COLOR(13,11,1)
2910 CALL SOUND(100,880,1)
2920 CALL KEY(1,KEY1,S)
2930 IF KEY1<>19 THEN 2920
2940 CALL COLOR(13,15,1)
2950 CALL HCHAR(10,3,137)
2960 CALL HCHAR(10,30,137)
2970 CALL HCHAR(12,3,128)
2980 CALL HCHAR(12,30,128)
2990 CALL COLOR(13,13,1)
3000 GOSUB 4840
3010 FOR I=9 TO 42
3020 CALL COLOR(I,2,16)
3030 NEXT I
3040 CALL KEY(1,KEY1,S1)
3050 CALL KEY(2,KEY2,S2)
3060 IF S1=-1 THEN 3040
3070 IF S2=-1 THEN 3040
3080 IF KEY1=19 THEN 3110
3090 IF KEY2=10 THEN 3190
3100 GOTO 3040
3110 CALL SOUND(300,-3,1)
3120 FOR M=1 TO 3
3130 CALL COLOR(2,11,11)
3140 CALL COLOR(2,7,7)
3150 NEXT M
3160 COL=3
3170 TEAM$="RED"
3180 GOTO 3260
3190 CALL SOUND(300,-2,1)
3200 FOR M=1 TO 3
3210 CALL COLOR(1,11,11)
3220 CALL COLOR(1,5,1)
3230 NEXT M
3240 COL=19
3250 TEAM$="BLUE"
3260 FOR I=9 TO 12
3270 CALL COLOR(I,16,16)
3280 NEXT I
3290 MSG$="OK_"&TEAM$&"_TEAM"
3300 GOSUB 4580
3310 C=13
3320 N=1
3330 CALL KEY(0,KEY,S)
3340 IF S<>1 THEN 3330
3350 IF KEY<51 THEN 3330
3360 IF KEY>57 THEN 3330
3370 KEY=KEY-50
3380 IF KEY<>VAL(SEG$(WORD$,
     N,1))THEN 3700
3390 CALL SOUND(100,880,1)
3400 KEY=KEY+64
3410 CALL HCHAR(9,C,KEY)
3420 IF N=LEN(WORD$)THEN 3460
3430 N=N+1
3440 C=C+2
3450 GOTO 3330
3460 FOR I=9 TO 12
3470 CALL COLOR(I,2,16)
3480 NEXT I
3490 CALL SOUND(100,330,1,392,1,524,1)
3500 CALL SOUND(500,330,1,392,1,524,1)
3510 MSG$="RIGHT"
3520 IF TEAM$="BLUE" THEN 3550
3530 COL=3
3540 GOTO 3560
3550 COL=19
3560 GOSUB 4580
3570 IF TEAM$<>"RED" THEN 3620
3580 RED=RED+1
3590 C=3
3600 SCORE=RED
3610 GOTO 3650
3620 BLUE=BLUE+1
3630 C=29
3640 SCORE=BLUE
3650 MSG$=STR$(SCORE)
3660 GOSUB 4640
3670 FOR I=1 TO 500
3680 NEXT I
3690 GO TO 1930
3700 CALL SOUND(100,110,1)
3710 IF TRY=2 THEN 1930
3720 TRY=TRY+1
3730 CALL HCHAR(9,13,95,13)
3740 IF TEAM$="RED" THEN 3780
3750 TEAM$="RED"
3760 COL=3
3770 GOTO 3800
3780 TEAM$="BLUE"
3790 COL=19
3800 MSG$="YOU_TRY_"&TEAM$
3810 GOSUB 4580
3820 GOTO 3310
3830 CALL SCREEN(2)
3840 FOR I=1 TO 16
3850 CALL VCHAR(1,I,40,24)
3860 NEXT I
3870 FOR I=17 TO 32
3880 CALL VCHAR(1,I,35,24)
3890 NEXT I
3900 FOR I=2 TO 4
3910 CALL HCHAR(I,2,95,4)
3920 CALL HCHAR(I,28,95,4)
3930 NEXT I
3940 FOR I=19 TO 21
3950 CALL HCHAR(I,2,95,14)
3960 CALL HCHAR(I,18,95,14)
3970 NEXT I
3980 CALL HCHAR(1,7,95,20)
3990 CALL HCHAR(2,7,95,20)
4000 FOR I=3 TO 7
4010 CALL HCHAR(I,8,103,19)
4020 NEXT I
4030 FOR I=8 TO 10
4040 CALL HCHAR(I,7,95,20)
4050 NEXT I
4060 CALL HCHAR(7,7,95)
4070 CALL VCHAR(3,7,98,4)
4080 IF CLEF=2 THEN 4110
4090 GOSUB 4340
4100 GOTO 4140
4110 R=3
4120 GOSUB 4480
4130 GOTO 4230
4140 IF CLEF<>3 THEN 4230
4150 FOR I=11 TO 15
4160 CALL HCHAR(I,8,103,19)
4170 NEXT I
4180 CALL HCHAR(16,7,95,20)
4190 CALL VCHAR(7,7,98,8)
4200 CALL HCHAR(15,7,95)
4210 R=11
4220 GOSUB 4480
4230 FOR I=7 TO 13
4240 CALL HCHAR(I,2,136,3)
4250 CALL HCHAR(I,29,136,3)
4260 NEXT I
4270 CALL HCHAR(8,3,128)
4280 CALL HCHAR(8,30,128)
4290 CALL HCHAR(10,3,137)
4300 CALL HCHAR(10,30,137)
4310 CALL HCHAR(12,3,137)
4320 CALL HCHAR(12,30,137)
4330 RETURN
4340 CALL HCHAR(2,9,96)
4350 CALL HCHAR(2,10,97)
4360 CALL HCHAR(3,9,99)
4370 CALL HCHAR(3,10,100)
4380 CALL HCHAR(4,8,101)
4390 CALL HCHAR(4,9,102)
4400 CALL HCHAR(5,8,104)
4410 CALL HCHAR(5,9,105)
4420 CALL HCHAR(5,10,106)
4430 CALL HCHAR(6,8,107)
4440 CALL HCHAR(6,9,108)
4450 CALL HCHAR(6,10,109)
4460 CALL HCHAR(7,9,99)
4470 RETURN
4480 CALL HCHAR(R,8,110)
4490 CALL HCHAR(R,9,111)
4500 CALL HCHAR(R,10,112)
4510 CALL HCHAR(R+1,8,113)
4520 CALL HCHAR(R+1,10,114)
4530 CALL HCHAR(R+2,9,115)
4540 CALL HCHAR(R+2,10,116)
4550 CALL HCHAR(R+3,8,117)
4560 CALL HCHAR(R+3,9,118)
4570 RETURN
4580 CALL HCHAR(20,3,95,12)
4590 CALL HCHAR(20,19,95,12)
4600 FOR I=0 TO LEN(MSG$)-1
4610 CALL HCHAR(R[illegible]OL+I,
     ASC(SEG$(MSG$,I+1,1)))
4620 NEXT I
4630 RETURN
4640 IF LEN(MSG$)=2 THEN 4660
4650 MSG$="_"&MSG$
4660 FOR I=1 TO 2
4670 CALL HCHAR(3,C,ASC(SEG$
     (MSG$,I,1)))
4680 C=C+1
4690 NEXT I
4700 RETURN
4710 FOR MMM=1 TO 11
4720 CALL HCHAR(R,6,97)
4730 CALL HCHAR(R-1,6,96)
4740 CALL SOUND(50,-7,5)
4750 CALL SOUND(MM-100,110,30)
4760 CALL HCHAR(R-2,8,97)
4770 CALL HCHAR(R-3,8,96)
4780 CALL SOUND(50,-7,5)
4790 CALL SOUND(MM-100,110,30)
4800 R=R-4
4810 NEXT MMM
4820 CALL SOUND(200,110,30)
4830 RETURN
4840 CALL SOUND(675,262,1)
4850 CALL SOUND(225,294,1)
4860 CALL SOUND(225,311,1)
4870 CALL SOUND(225,110,30)
4880 CALL SOUND(225,262,1)
4890 CALL SOUND(225,110,30)
4900 RETURN
4910 FOR J=12 TO 20 STEP 2
4920 CALL HCHAR(J,2,119,30)
4930 NEXT J
4940 FOR I=F1 TO F1+8 STEP 2
4950 READ MSG$
4960 FOR J=0 TO LEN(MSG$)-1
4970 CALL HCHAR(S1-I,I+J,ASC(SEG$
     (MSG$,J+1,1)))
4980 NEXT J
4990 NEXT I
5000 FOR I=F2 TO F2+6 STEP 2
5010 READ MSG$
5020 FOR J=0 TO LEN(MSG$)-1
5030 CALL HCHAR(S2-I,I+J,ASC(SEG$
     (MSG$,J+1,1)))
5040 NEXT J
5050 NEXT I
5060 MSG$="PRESS ANY KEY"
5070 GOSUB 4600
5080 CALL KEY(0,K,S)
5090 IF S=0 THEN 5080
5100 RETURN
5110 DATA " E - EVERY "," G - GOOD "," B - BOY "," D - DOES "," F - FINE "
5120 DATA F,A,C,E
5130 DATA " G - GOOD "," B - BOYS "," D - DO "," F - FINE "," A - ALWAYS "
5140 DATA " A - ALL "," C - COWS "," E - EAT "," G - GRASS "
5150 FOR I=3 TO 30
5160 CALL VCHAR(1,I,32,24)
5170 NEXT I
5180 RETURN
```

# DEALER DIRECTORY

## Denver, CO

Your headquarters for Texas Instruments TI-99/4A Personal Computers, software and peripherals is CoxCo Personal Computer Center. We maintain a complete stock of Command Modules and related equipment for TI and Atari Personal Computers. **Coxco Personal Computer Center, 12351 West 64th Ave., Arrada, CO 80004, (303) 421-6361.**

## Waterloo, IA

Texas Instruments Large selection of Accessories and Software for 99/4 at Dhein's True Value, 7 W. Airline Hwy, Waterloo, IA 50701. Sale prices at store are also given on mail orders. Write for price list. **Dhein's True Value, 7 W. Airline Highway, Waterloo, IA 50701, (319) 232-6225.**

## Lexington, KY

A complete line Texas Instruments dealer. We stock all Texas Instruments calculators and accessories, learning aids, home computer products, accessories, peripherals and software. Atari computers and Hewlett Packard calculators in stock. **CBM Incorporated, 198 Moore Drive, Lexington, KY 40503, (606) 276-1519, (800) 354-9099.**

## Louisville, KY

Serving the Kentuckiana area with the full line of TI-99/4A computers, accessories and software including custom business applications. All TI calculators, DS990 minicomputer series and the new Business System 200 series. **Corporate Computer Services, Inc., Suite 455, 10170 Linn Station Road, Louisville, KY 40223, (502) 423-9187.**

## Springfield, MO

A full-line Texas Instruments TI-99/4A Computer dealer—Large selection of peripherals and software—Custom programming for the TI-99/4A—Complete price list on request. **G & G Data Processing, 209 E. Walnut, Springfield, MO 65806, (417) 869-1489.**

Call a Dealer Today

## Marlow Heights, MD

Specializing in Texas Instruments 99/4 computer and accessories. We have a large selection of Software for the TI-99/4, TRS-80, Apple, and Atari. We also stock Lobo Drives and Epson Printers. **Wills Computer Store, 4410 Stamp Road, Marlow Heights, MD 20031, (301) 423-4525.**

## Minneapolis, MN

Authorized Texas Instruments TI-99/4 computer dealer offering you the best prices in the U.S.A. on the computer and all peripherals. Also, a wide variety of printers. CRT's available at lowest prices. No collect calls, please. **Calculators, Inc., 7409 Fremont Avenue South, Minneapolis, MN 55423, (612) 866-8908.**

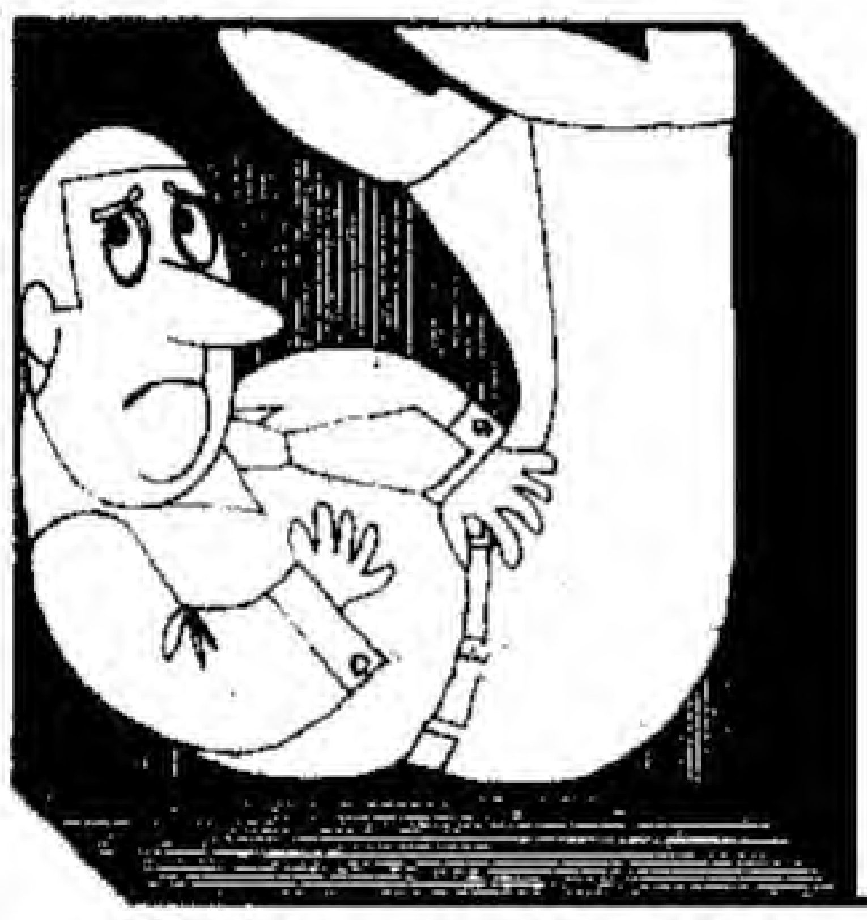
## New York, NY

"Your Personal Guide to Personal Technology." We carry the complete TI-99/4 System. For demonstration of the TI-99/4 at home, school or office, or more information call Next Tech, Inc. today. **Next Tech, Inc., 350 Fifth Avenue, Suite 3308 New York, NY 10001, (212) 792-8768.**

## Armonk, NY

Wide assortment of TI software, hardware and accessories all at discount prices. Also authorized Scott, Foresman dealer. Write or phone for nine-page catalog with hundreds of items. **Microcomputers Corporation, 34 Maple Avenue, Armonk, NY 10504, (914) 273-6480.**

NEED MORE ROOM FOR YOUR MESSAGE?



TRY A 99'er SPACE AD

Call or Write TODAY

## Minneapolis, MN

Authorized TI-99/4A & Scott, Foresman dealer for educational electronic materials in the 5-state area. We have qualified educators to help with your educational microcomputer needs. We are also a full sales, service and leasing organization. **Tele-Terminals, Inc., 7216 Boone Ave. North, Brooklyn Park, MN 55428. (612) 535-5330.**

## Great Falls, MT

We've got the best prices because we want your business. Complete line of TI products and Epson printers. Monthly SALES on all items. Call now or write: **IRCA Inc., N.E. Bypass Hwy. 87, P. O. Box 6125, Great Falls, MT 59406, (406) 727-8835.**

## Olean, NY

Texas Instruments TI-99/4 Home Computer Systems, software, hardware, and accessories at discount prices. APF-Imagination Machine. Video Sales and Service. **BLOISE Electronics, 1018 West State Street, Olean, NY 14760. (716) 372-1493.**

## Dayton, OH

The only full line TI distributor stocking Semiconductors, DSG (700-800 OMNI Products) TM990 Microcomputer Boards, Calculators, complete in depth inventory including 99/4 Products-Software. Ohio 1-800-762-9510. Surrounding area 1-800-543-9550. Will ship. Call us. **Esco Inc., P. O. Box 1166, 221 Crane St. Dayton, OH 45403, (513) 226-1133**

## Portland, OR

Friendly pre-programmed computing systems for the office and home. Featuring T. I. 99/4A's, Hewlett Packard, Xerox, IBM, and most printers. With T. I. 99, discounted packages of all configurations. 24 Hour Home Computer Hotline. **The Electronic Cottage, 2336 SW Osage, Suite 407, Portland, OR 97205, (503) 224-9282.**

## Tacoma, WA

Specializing in Texas Instruments Computers all Hardware and software. Also specializing in Satellite Earth Station, Video Recorders, and TV, will install Earth Stations anywhere in the United States, very low prices we want your business. **VIDSAT, 11002 Pacific Ave., Tacoma, WA 98444.**

## Green Bay, WI

We specialize in the Texas Instruments 99/4 computer. We handle a complete accessory line for it, at discount prices. Custom programming for the 99/4 also done. Write for a complete price list. **The Micro House, 527 Simonet Street, Green Bay, WI 54301, (414) 432-2871.**

## Washington, D. C.

Marinchip systems 9900 and Z800 dealer. Networking software and CPM interface. Technico experience in-house. Epson printers, televideo CRT's, cables, TI 99/X to S-100 bus card available system customization. Hard disk, tape. Any AMP connectors. **Interface Technology, P O Box 745, College Park, MD 20740, (301) 490-3608.**

## RATES

**Listings here are $30 per issue; minimum insertion, 3 issues (6 months). Prepayment of $90 required. Ads include 35 words describing products and services, plus company name, address and phone. (No merchandise prices, please.) Call Pat at 503-485-8796 or write 99'er Magazine, Ad Department, 2715 Terrace View Drive, Eugene, OR 97405.**

**ATTENTION TI DEALERS**

Did You Know That You Can Be Reimbursed For Advertising Texas Instruments Products In This Magazine?

Call or Write

99'er Magazine/Ad. Dept.
2715 Terrace View Drive, Eugene, OR 97405
Tel. (503) 485-8796

# New from HIPLØT™ multi-pen plotting for as little as $1480*.

The new HIPLØT DMP Series 6-pen option makes high performance multi-pen plotting affordable. It's available on the DMP 2, 3, and 4 models in the HIPLØT family so you can enjoy the advantages of multi-colored plots on 8½" × 11" (DIN A4) surfaces. Of course, you also get the standard HIPLØT range of capabilities such as intelligence, controls, interfaces and resolutions. There's a model for virtually every plotting application.

**Big Performance in a Small Plotter**

Since it's introduction, the HIPLØT DMP Series has been recognized as the innovative plotter line which made low-cost, high performance digital plotting a reality.

Now, with our new 6-pen option, there's an exciting new dimension in the DMP Series' versatility. Imagine two standard models with RS-232-C and parallel interfaces, four intelligent models with RS-232-C or Centronics® compatible interfaces, a choice of controls, resolutions, and pen speeds. Add to this the ability to plot with 6-pens on paper, vellum or mylar (ideal for overhead projectors) and you have the ultimate plotter price/performance combination — the perfect choice for the user or OEM.

**8-Pen Models Also Available.**

If you need a little more capability, take a look at our new 8-pen option. It's available on the DMP 5, 6, and 7 so you can have 8-pen multi-colored plots on 11" × 17" (DIN-A3) surfaces.

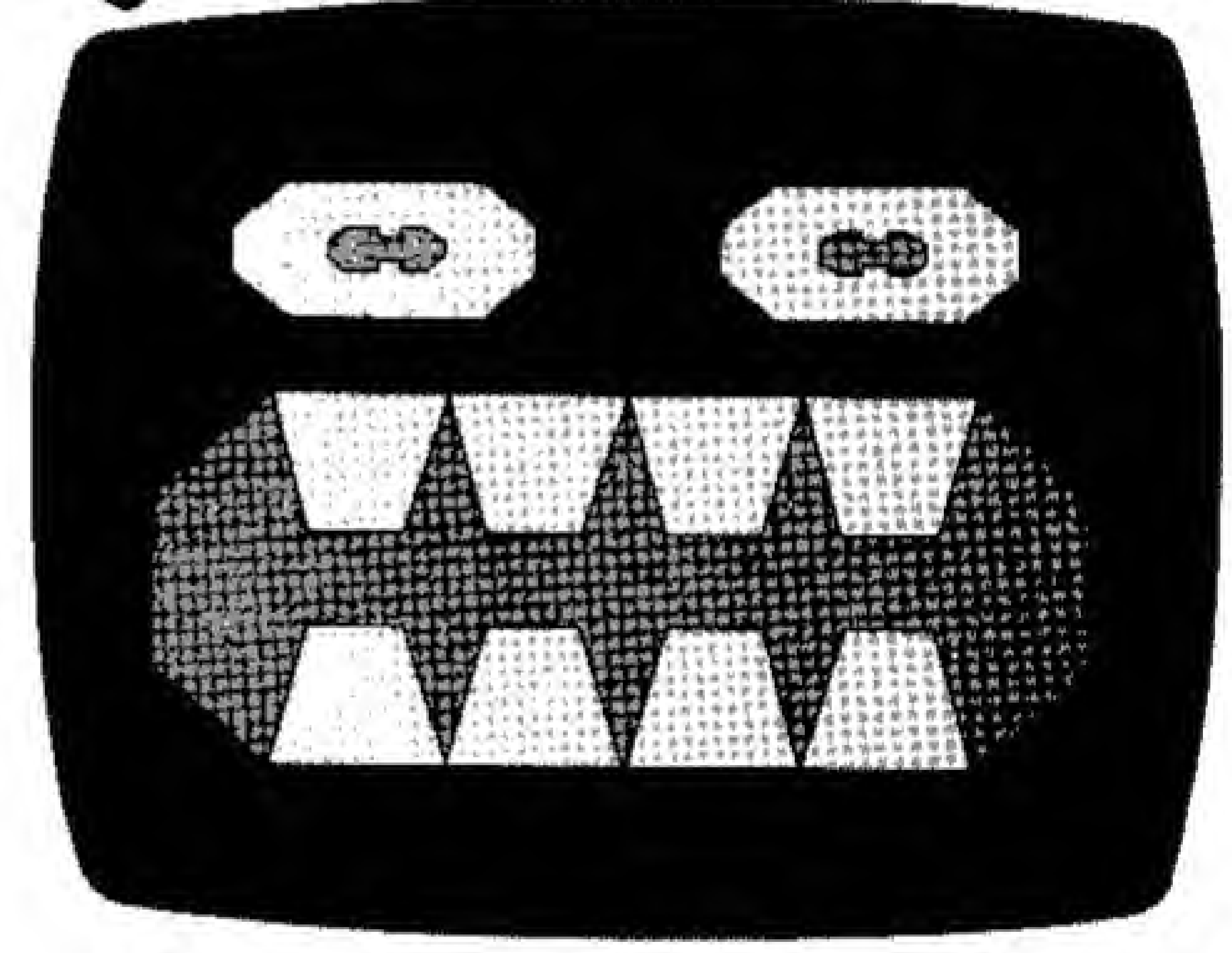
*Why wait? Let us send you complete information on this breakthrough in* **affordable, multi-pen plotting.** *Contact Houston Instrument, P.O. Box 15720, Austin, Texas 78761. (512) 835-0900. For rush literature requests, outside Texas, call* **toll free 1-800-531-5205.** *For technical information ask for operator #5. In Europe contact Houston Instrument, Rochesterlaan 6, 8240 Gistel, Belgium. Phone 059/27-74-45.*

INSTRUMENTS & SYSTEMS DIVISION
Together...we'll create tomorrow.

**BAUSCH & LOMB**

TM Trademark of Houston Instrument.
* U.S. Suggested retail prices.
Centronics® registered trademark of Centronics Data Computer Corp.

# THE MONSTER IS FREE!



## During the Texas Instruments Home Computer Free Software Offer.

The Monster, otherwise known to his friends as "Munchman", is one of the most fun and exciting games Texas Instruments has ever had. And now, he can be yours at no cost whatsoever during our Texas Instruments Home Computer Free Software Offer. Here's how it works:

If you buy any four Texas Instruments Solid State Software™ modules, or a Texas Instruments software album (three modules in a convenient storage package) between February 1 and May 15, 1982, we'll send you the exciting new "Munchman Arcade Game" absolutely free. The modules you choose can be from any category: Home Finance, Children's Education or Entertainment. And the Munchman will be on us.

Munchman is an offer worth $39.95, but with a deal like this, people are going to eat it up. So hurry. Take the coupon in this ad to your participating retail outlet... and set the monster free.

TEXAS INSTRUMENTS
INCORPORATED

Mail to:
"Munchman" Offer
P.O. Box 725, Dept. HC-HBO,
Lubbock, Texas 79491

Yes, I want to set the Monster free. I am enclosing a proof-of-purchase with this coupon. Please send "Munchman" to:

| | | | |
|---|---|---|---|
| Name | | | |
| Address | | | |
| City | State | Zip | Phone |
| Serial # of TI 99/4 Console | Age of Buyer | Occupation | |
| No. of children | Ages | Name of store where purchased | |
| Intended Use | Home | Personal Finance | Education |

```
630 CALL LOAD (Y-17, ASC (S$))
640 NEXT Y
650 CALL LOAD (-17, LEN (PRG$))
660 RUN "DSK1.ABCDEFGHIJ"
```

The lines from 610 to 640 overwrite the constant in 660. In addition, line 650 adjusts the length of the constant as necessary. This code assumes that line 660 is the first line in memory. To accomplish this the easiest method is to type line 660 in first. If you already have most of your load program typed in, leave 660 out or erase it, SAVE the program in MERGE format to disk, type NEW, then line 660, then MERGE the rest of the program back in.

For those who don't like the preceding way of finding the right line, the following program should suggest another way:

```
100 REM LIST STATEMENT NUMBERS
    AND LINE LENGTHS
110 REM REQUIRES EXTENDED BASIC
    AND MEMORY EXPANSION
120 DEF MA(X)=X+65537*(X>=32768)
130 CALL PEEK (-32718,X,Y)
140 FIRSTPOINTER=MA(X*256+Y)
150 PRINT "STATEMENT NO
    LENGTH"
160 FOR I=FIRSTPOINTER -2 TO -33000
    STEP -4
170 CALL PEEK (I, A, B, C, D)
180 IF A+B=0 THEN 220
190 CALL PEEK (MA(C*256+D),L)
200 PRINT A*256+B, L
210 NEXT I
220 PRINT "PROGRAM LENGTH IS ";
    ABS(I)
999 END
```

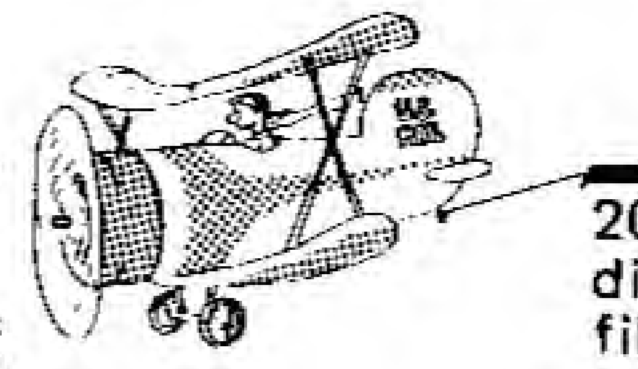
This program shows how the statements are located through BASIC. The address generated in 190 for each statement points to the text of the statement. Each line is stored as a length, then the bytes indicating tokens for BASIC keywords or variable names or constants, then a zero to indicate end-of-line.

The preceding information should keep avid users busy for a while. One thing you'll find is that the constant 1 takes three bytes to represent (200, 1, 49 where the 200 indicates a numeric constant, the 1 is the length and 49 is the ASCII code for 1), while a variable set to 1 can be represented in one byte if it's a single letter. Of course this will cause some overhead (about 20 bytes) but at a saving of two bytes for every 1 removed from your programs, you may just be able to fit that one more feature into memory. Most large programs can save hundreds of bytes this way.

Tim MacEachern
Dartmonth, Nova Scotia, CANADA

*Thanks for your method, Tim. Readers can add this to the alternate solution presented by John Clulow in these pages last issue. For the convenience of readers who would like to try Tim's method, we present a consolidated version below.*

```
1 REM ********************
2 REM * GENERAL PURPOSE *
3 REM * PROGRAM LOADER  *
4 REM * BY C.M.EHNINGER *
5 REM ********************
6 OPTION BASE 1
7 DIM PG$(20)
8 IMAGE ##
9 CALL CLEAR
10 DISPLAY AT(12,6)ERASE ALL:"DISK? (1-3): 1";
11 ACCEPT AT(12,19)SIZE(-1)VALIDATE("123"):D$
12 OPEN #1:"DSK"&D$&".",INPUT ,RELATIVE,INTERNAL
13 INPUT #1:N$,A,A,A
14 DISPLAY AT(1,8)ERASE ALL:"DSK"&D$&" - "&N$;
15 I=0
16 FOR X=1 TO 20
17 I=I+1
18 IF I>127 THEN 42
19 INPUT #1:P$,A,B,B
20 IF LEN(P$)=0 THEN 26
21 IF ABS(A)<>5 THEN 19
22 DISPLAY AT(X+2,10):USING 8:X;
23 DISPLAY AT(X+2,14):P$;
24 PG$(I)=P$
25 NEXT X
26 DISPLAY AT(X+2,10):USING 8:X;
27 DISPLAY AT(X+2,14):"TERMINATE";
28 DISPLAY AT(X+3,14):"CHOICE? 1";
29 ACCEPT AT(X+3,22)SIZE(-2)VALIDATE(DIGIT):K
30 IF K=X THEN END
31 IF K<1 OR K>20 THEN 28
32 IF LEN(PG$(K))=0 THEN 28
33 CLOSE #1
34 CALL CLEAR
35 REM ************************
36 REM *    RUN STATEMENT     *
37 REM *       MODIFIER       *
38 REM *                      *
39 REM * BY  TIM MACEACHERN  *
40 REM *      DARTMOUTH       *
41 REM *     NOVA SCOTIA      *
42 REM *                      *
43 REM *  SAVE THE PROGRAM *
44 REM * WITH MERGE OPTION *
45 REM * WITHOUT LINE 53.   *
46 REM * ENTER NEW, ENTER   *
47 REM * LINE 53.  THEN     *
48 REM * MERGE PROGRAM.     *
49 REM ************************
50 REM
51 CALL INIT :: FOR Y=1 TO 16 :: S$=SEG$
   ("DSK"&D$&"."&PG$(K)&RPT$(CHR$(0),10),Y,
   1):: CALL LOAD(Y-17,ASC(S$)):: NEXT Y
52 CALL LOAD(-17,5+LEN(PG$(K)))
53 RUN "DSKN.ABCDEFGHIJ"
```



## What? Another elegant solution?

Dear Sir:

In the July/August 1981 issue of "99'er" (Vol. 1, No. 2) Letters to the Editor column Mr. Charles Ehninger expressed his frustration with not being able to implement a menu program which would display the programs on a diskette, allow a selection, and then RUN that program without the menu program being tailored to each diskette. His main problem was that TI Extended BASIC requires a literal and will not accept a variable as the object of the RUN command.

I believe that I have solved this problem, albeit with some code which requires not only TI Extended BASIC but also the 32K RAM Memory Expansion, is (HORRORS!) self-modifying, and forever linked to the current version of TI Extended BASIC (I hear rumors of a new, improved model which may not use the same constructs I depend on for this program). By perusing the accompanying listing, you may discover that I have solved another problem, namely, that although TI implemented lower case in the BASIC resident in the new TI-99/4A, they have not done so for TI Extended BASIC yet (more rumors?). In fact, your readers may have difficulty entering the program unless they first enter line 170 and run it as a one-liner.

With a TI-99/4A, there are two pre-requisites for entering lower case into TI Extended BASIC programs: 1) the lower case character set must be defined (the regular BASIC does this automatically), and 2) the keyboard must be reconfigured to enable lower case character entry (WARNING: my method also passes CONTROL and FUNCTION characters to TI Extended BASIC if they are inadvertantly entered, and may cause unpredictable results!). Requirement 1) is met by pulling the upper case character definitions into a string, modifying them, and installing them as the definitions of the lower case ASCII set (characters 97 to 122). Requirement 2) is met by the CALL KEY statement specifying keyboard 5 (the two other parameters are not used but required by the syntax of the statement). These steps also work on a TI-99/4 and will define the lower case (although the characters are not as well-defined as on the 4A) but the CALL KEY (5, I, I) will be ignored and there is no provision in the old keyboard for entering lower case.

Once line 170 has been entered and run, you may then use your 4A to enter the remaining lines (the string in line 160 MUST be in lower case, line 330 contains 15 spaces and MUST be the last line in the program, regardless of its line number!). You will notice many similarities to Mr. Ehninger's program, and I thank him for the format. The last statement in line 150 simply checks if the lower case set has already been defined and skips over that section if it has, saving execution time. Line 160 prints the lower case character set on the screen so you can watch it being defined (it also avoids a lot of time watching a blank screen!). Line 190 allows selection of DSK1, 2, or 3, and line 200 brings in the name of the selected diskette. Lines 210 through 270 bring in the file names from the disk, reject them if they are not PROGRAM files, and construct the menu on the screen. I retained Mr. Ehninger's limit in line 210 which makes the menu only one screenful of the first 20 programs on the diskette. Your readers may want to modify the program so that it lists all programs on multiple screens; the program may be modified so long as the RUN statement is the highest numbered line. The last item in the menu is always "Terminate", which just exits the program (line 280).

The statements in line 310 do the work of RUNning the selected program (whose name is in PG$(K)) by modifying the last line of the menu program (currently line 330—the 15 spaces make room for the longest possible program name). The CALL PEEK and CALL LOAD statements only work with Expansion RAM after a CALL INIT, and a TI Extended BASIC program will reside in Expansion RAM if it is installed. Thus, if we can find the program in memory, we can modify it. One of the things that happens when a program is entered into BASIC is the generation of a Line Number Table in memory. This table is necessary because the lines of a BASIC program are not necessarily in order in memory (they can be entered in any order, and they can be edited which scrambles their physical order). The line number table for a TI Extended BASIC program in Expansion RAM is a sequential list by line number of all the numbered lines of a program, and the BASIC interpreter keeps a pointer (a two-byte address) to the location of the highest used line number (and hence the last entry in the Line Number Table) at location -31952 (that's really 8330 in Hexadecimal, and 33584 in Decimal, but the PEEK statement needs an integer decimal address in the range of -32767 to +32767, so to convert we subtract 65536 from any value above 32767). We now bring in this pointer's two bytes as A and B. You can get the decimal value of a two-byte hex number by the formula A*256+B. The next CALL PEEK uses the pointer's value to access the line number table entry for the highest line number (remember, this number is greater than 32767, so subtract 65536). Since the Line Number Table contains two entries per BASIC line, its line number as a two-byte value followed by the two-byte address of the line in memory, and since we want the address only, we add 2 to the pointer and bring the address into A and B. BASIC stores program lines in a TOKENIZED format, which means that reserved words such as RUN take up only one byte of storage. In memory, line 330 is stored as a RUN token, a token which announces a quoted string, a byte which gives the length of the string, the characters of the string, and a zero byte which is the string terminator. Since we are going to modify the string, we need to compute its length and poke that value into the length byte (CALL LOAD is a POKE). The location of the length byte is computed from the start of line address we just PEEK'ed, the LEN of the string to replace the one in line 330 is obtained, and the CALL LOAD updates the length byte.

Now we must poke the new string, consisting of "DSK", the selected drive, " . ", and the name of the selected program, into the line one character at a time and terminate the string with a zero byte. Once the line has been modified, it will contain a RUN statement with a literal object which is in the form of "DSK2.MYPROG". You may observe this action by running the menu program and opening the disk drive door before making a program selection. The RUN statement will fail with an I/O ERROR and

Continued on p. 94

# 99'er TI Fest is Coming... to San Francisco

**ATTEND** Applications Seminars for Using Personal Computers in Business, Science, Engineering, and for Home Management & Financial Decision-Making!

**SEE & OPERATE** the Latest Personal Computer Hardware & Software Products from TI & Third Parties!

**ATTEND** Programming Language Tutorial Seminars in TI BASIC, Extended BASIC, LOGO, TMS9900 Assembly Language, UCSD Pascal, & PILOT!

**WATCH** Demonstrations of TI's State-of-the-Art Technology & Catch a Glimpse of Products of the Future!

**PLAY** Spectacular Personal Computer Arcade Games . . . . Enter the Competition to Win Computers, Peripherals, and Software!

**WITNESS** Unique Applications for TI Electronic Learning Aids & Innovations in Computer Assisted Instruction!

**MEET** with Users Groups and Share Ideas with Others of Similar Interest!

**SAVE** Money by Purchasing the Products You Need at Special Show Discounts!

**PARTICIPATE** in the Users Forum to Voice Your Opinions & Ask Questions Directly to TI!

**LEARN** How to Profitably Produce Software and Hardware for this Dynamic, New Mass Market!

**A UNIQUE OPPORTUNITY** for Dealers, Manufacturers, & Product Developers to Display & Sell Directly to Thousands of Eager Buyers!

**HUNDREDS** of Valuable Door Prizes!

**RIDE** a Cable Car, Sample the Delicacies of China Town & Fisherman's Wharf, and Cross the Famous "Golden Gate"!

**TRANSPORTATION**, Sightseeing, and Hotel Discount Packages Available!

And **MUCH, MUCH, MORE!**

**SAN FRANCISCO**

October 1982

**DON'T MISS IT!**

☞ See the Many Reasons Why the Versatile Texas Instruments Personal Computers have become the Value & Innovation Leaders.

99'er TI Fest

A Personal Computer Conference, Trade Show & Seminar
Presented by 99'er Magazine

**Watch For Additional Details in Next Issue.**

99'er Magazine • 2715 Terrace View Drive • Eugene, OR 97405 • Tel. (503) 485-8796

99'er magazine™ ON TAPE

## Selected Programs from 99'er magazine™ Are Now Available on Cassette Tape As A Service to Our Readers.*

**PACKAGE #E1**
**TI BASIC**

Typing for Accuracy (Vol 1, No. 3)
Let's Learn Notes (Vol 1, No. 3)
Mystery Words (Vol. 1, No. 4)
Homework Helper: Fractions (Vol. 1, No. 1)
Homework Helper: Division (Vol. 1, No. 4)

**$12/sub; $20/non-sub**

**PACKAGE #U1**
**TI BASIC**

Electronic Home Secretary (Vol. 1, No. 2)
Micro Bartender (Vol 1, No. 4)
Interactive Forms Generator (Vol. 1, No. 3)
Music Text Editor & File Player (Vol 1, No. 1)

**$10/sub; $16/non-sub**

**PACKAGE #G1**
**TI BASIC**

Space War (Vol. 1, No. 3)
Battle at Sea (Vol. 1, No. 2)
Maze Race (Vol. 1, No. 4)
Anti-Aircraft Gun (Vol. 1, No. 1)
Invasion from Space (Vol. 1, No. 1)
Enemy Attack (Vol. 1, No. 1)

**$15/sub; $25/non-sub**

**PACKAGE #G2X**
**Extended BASIC**

Interplanetary Rescue (Vol. 1, No. 4)
Dogfight (Vol. 1, No. 3)
Sprite Chase (Vol. 1, No. 2)
Dodge'em (Vol. 1, No. 4)

**$10/sub; $16/non-sub**

- All purchasers of these package are responsible for obtaining the individual documentation and program instructions in the indicated back issues of the magazines. When an issue goes out-of-print, copies of the program articles will be made available for purchase.
- If programs have been updated/enhanced since original publication, a printed copy of documentation changes will be provided with the tapes.
- Both subscriber & non-subscriber prices are indicated. To be eligible for the lower subscriber prices, you must be a current, paid magazine subscriber. All orders will be verified prior to shipment.
- Authors of these and future programs distributed on tape will receive Bonus Payments based on the number of tapes sold; so in fairness to them, please observe the Copyright laws, and report any incidents of "piracy" to our office.
- Recognized TI User Groups should contact us for special bulk terms & rates. No dealers, please.

* Not for Duplication or Resale. This service will exist only as long as it fulfills a useful function and isn't abused.

Use the bind-in card in the back of the magazine for your convenience in ordering.

Available for Shipment March 31, 1982

SHIPPING – $1.00 for 1st TAPE, 50¢ ea. add. Foreign Orders Shipped Airmail – $3.00 for 1st TAPE, 75¢ ea. add.

## Back Issues of 99'er magazine™ are Still Available . . . but quantities are limited

**so ORDER TODAY!** Each Only $3.95 postpaid

**ISSUE #1 (Partial Contents):**
•How To Write Your Own Programs •An External Keyboard •Epson MX-80 & Printer Graphics •Space & Combat Games in TI BASIC •Text-to-Speech •TMS9900 Machine & Assembly Language •Planning & Forecasting in BASIC •UCSD Pascal & Third-Party Development Systems •Power Line Problems •LOGO & Lamplighter •Music Text Editor •TM990/189 University Module •Homework Helper: Fractions •Computer Chess •and much, much more.

**ISSUE #2 (Partial Contents):**
•Information Utilities and the Electronic Cottage •Data Communications •Marinchip Systems M9900 •Software Conversion: TRS-80 to TI BASIC •Small Investor and the TI-99/4 •The Electronic Home Secretary •Typing Tutor •TI LOGO and the Space Shuttle •Civil Engineering Fundamentals •Catch & Match Games in BASIC & Extended BASIC •Bombs Away on the University Module •Interfacing a Digitizer •and much, much more.

**ISSUE #3 (Partial Contents):**
•Interactive Forms Generator •Business Software Reviews •A BASIC Program that Writes BASIC Programs •Applesoft to TI BASIC •Games of Agility & Derring—Do •Pascal Development System •Personal Record Keeping •Adding Your Own Disk Drives •Preliminary Look at TI's Editor/Assembler •Memory & I/O plus EPROM Programmer for University Board •State of the Art in Computer Assisted Instruction •Precautions & Programming Ideas for Beginners •and much, much more.

# Index to Advertisers

| | | | |
|---|---|---|---|
| A. Ace Computer | 3 | Komputar Works | 75 |
| American Software Design & Distribution Co. | 28 | Linear Aesthetic Systems | 32 |
| Anthistle Systems and Programming Ltd. | 74 | Microcomputers Corporation | 29 |
| The Bach Company | 12 | The Micro House | 62 |
| CBM Inc. | 18 | Norton Software | 28, 77 |
| Compu Tech | 77 | Olympic Sales Co. | 15 |
| Computer Connection | 62 | PRP Computer Graphics | 14 |
| Computer—Ed. | 30 | Pablo Diablo | 62 |
| Computer Mail Order | 55 | Pewterware | 30 |
| Cumberland Technology | 77 | Pike Creek Computer Co., Inc. | 32 |
| Data Systems | 77 | RCL Computers | 78 |
| Denali Data | 56 | RKS Enterprises, Inc. | 60 |
| Eastbench Software Products | 62 | Scott, Foresman and Co. | 46 |
| Ehninger Associates, Inc. | 35 | The Software Exchange Group | 14 |
| Electronic Specialists Inc. | 29 | Sunshine Software | 32 |
| Elek - Tek, Inc. | 38 | Tam's Inc. | 2 |
| Epson America, Inc. | 20, 21 | Texas Instruments, Inc. | 90, 96 |
| Extended Software Co. | 61 | Texas Instruments Supply | 16 |
| FFF Software | 30 | Unisource Electronics, Inc. | 22, 23 |
| Fantasy Computing | 14 | 99'er Bookstore | 80, 81 |
| George Goode and Associates Inc. | 30 | 99'er Magazine | 18, 37, 47, 93 |
| Hardin's Computer Solutions, Inc. | 54 | 99'er TI-Fest | 92 |
| Houston Instrument | 89 | 99'er ware | 94, 95 |
| International Home Computer Users Assn. | 13 | | |

## Letters . . . from p. 91

you may list the last line to see the magic. WARNING: SAVE YOUR MENU PROGRAM BEFORE RUNNING IT!

Two final comments: 1) I think "99'er" is fantastic! Keep up the good work! 2) I have a reputation to protect in the computer industry which means that I am proud to own a TI-99/4A and to program in TI Extended BASIC, but which also means that on code such as this, I must sign myself,

A. Kludge
(Postmarked) Van Nuys, CA

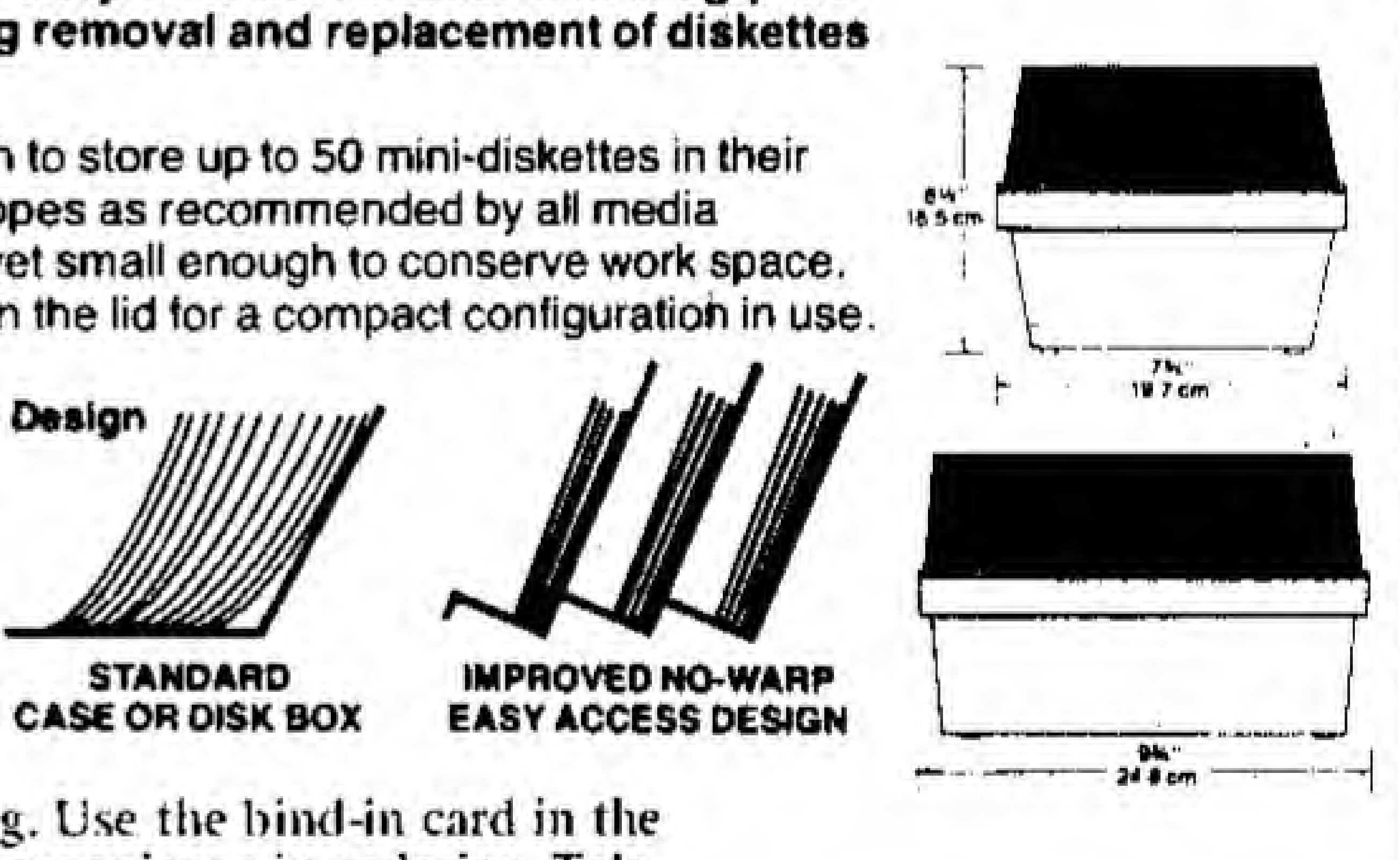
```
100 REM ********************
110 REM * GENERAL PURPOSE *
120 REM *  DISKETTE MENU  *
130 REM *  BY  A. KLUDGE  *
140 REM ********************
150 OPTION BASE 1 :: DIM PG$(20):: CALL CLEAR :
    : CALL CHARPAT(97,A$):: IF SEG$(A$,7,8)=
    "38447C44" THEN 180
160 DISPLAY AT(15,2):"ABCDEFGHIJKLMNOPQRSTUVWXYZ"
170 FOR I=65 TO 90 :: CALL CHARPAT(I,A$):
    : B$="0000"&SEG$(A$,1,4)&SEG$(A$,7,4)&SEG$
    (A$,13,4):: CALL CHAR(I+32,B$):: NEXT I :
    : CALL KEY(5,I,I)
180 IMAGE ##
190 DISPLAY AT(1,9)ERASE ALL:"DISKETTE MENU" :
    : DISPLAY AT(12,6):"DISK? (1-3): 1" :
    : ACCEPT AT(12,19)SIZE(-1)VALIDATE("123"):D$
200 D$="DSK"&D$&"." :: OPEN #1:D$,INPUT ,
    RELATIVE,INTERNAL :: INPUT #1:N$,A,A,A
    :: DISPLAY AT(1,8)ERASE ALL:SEG$(D$,1,3)&
    "    "&N$;:: I=0
210 FOR X=1 TO 20 :: I=I+1 :: IF I>127
    THEN K=X :: GOTO 280
220 INPUT #1:P$,A,B,B
230 IF LEN(P$)=0 THEN 260
240 IF ABS(A)<>5 THEN 220
250 DISPLAY AT(X+2,10):USING 180:X :: DISPLAY AT
    (X+2,14):P$ :: PG$(X)=P$ :: NEXT X
260 DISPLAY AT(X+2,10):USING 180:X :: DISPLAY AT
    (X+2,14):"TERMINATE" :: DISPLAY
    AT(X+4,2):"CHOICE? 1"
270 ACCEPT AT(X+4,10)SIZE(-2)VALIDATE(DIGIT):K
280 IF K=X THEN CALL CLEAR :: CLOSE #1 :: END
290 IF K<1 OR K>20 OR LEN(PG$(K))=0 THEN 260
300 CLOSE #1
310 CALL INIT :: CALL PEEK(-31952,A,B):: CALL
    PEEK(A*256+B-65534,A,B):: C=A*256+B-65534 :
    : A$=D$&PG$(K):: CALL LOAD(C,LEN(A$))
320 FOR I=1 TO LEN(A$):: CALL LOAD(C+I,ASC(SEG$
    (A$,I,1))):: NEXT I :: CALL LOAD(C+I,0)
330 RUN "DSKX.1234567890"
```

# FAST and EASY
## Retrieval of the Diskette YOU NEED . . .

**. . .While Guarding Against Data Dropout Caused By Improper Storage and Handling.**

**The MINI-FLEX is styled with low sides and snag-proof edges . . . making removal and replacement of diskettes easy and safe.**

Spacious enough to store up to 50 mini-diskettes in their protective envelopes as recommended by all media manufacturers, yet small enough to conserve work space. The base nests in the lid for a compact configuration in use.


Mini-Flex Partition Design
STANDARD CASE OR DISK BOX
IMPROVED NO-WARP EASY ACCESS DESIGN

Partitions: 3
Color: Smoke and Sand
1 Package MINI-FLEX Guides
Shipping Weight: 3 Lbs./1.35 Kg.

**MINI-FLEX DISKETTE FILE**

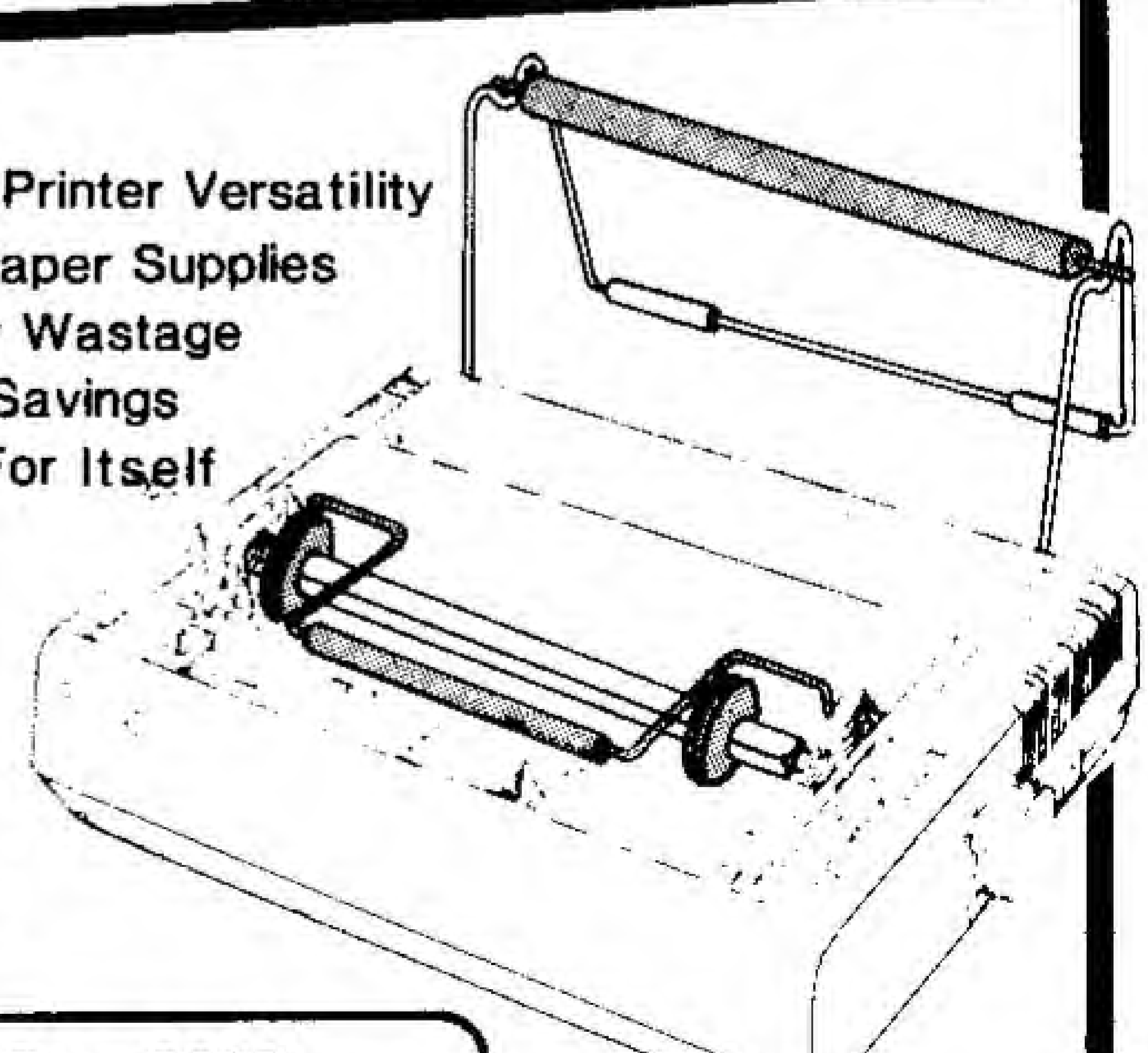
List Price: $29.95

☆ **Special Price $24.95**

Add $2.50 shipping and handling. Use the bind-in card in the back of the magazine for your convenience in ordering. Telephone orders accepted if charged to credit cards.

# Single-Sheet & Roll-Feed – A Cost-Effective Upgrade For Your EPSON MX-80 Printer

- Increased Printer Versatility
- Cheaper Paper Supplies
- Less Paper Wastage
- Large Time Savings
- Quickly Pays For Itself



Each Kit Comes Complete With:
- 2 Acrylic Drive Wheels (With Rubber Friction Rings) Machined to Exacting Tolerances (+ .0005, - 0.0 inches) for a Snug Fit
- 1 Weighted Pinch Roller Assembly
- 1 Roll Paper Holder

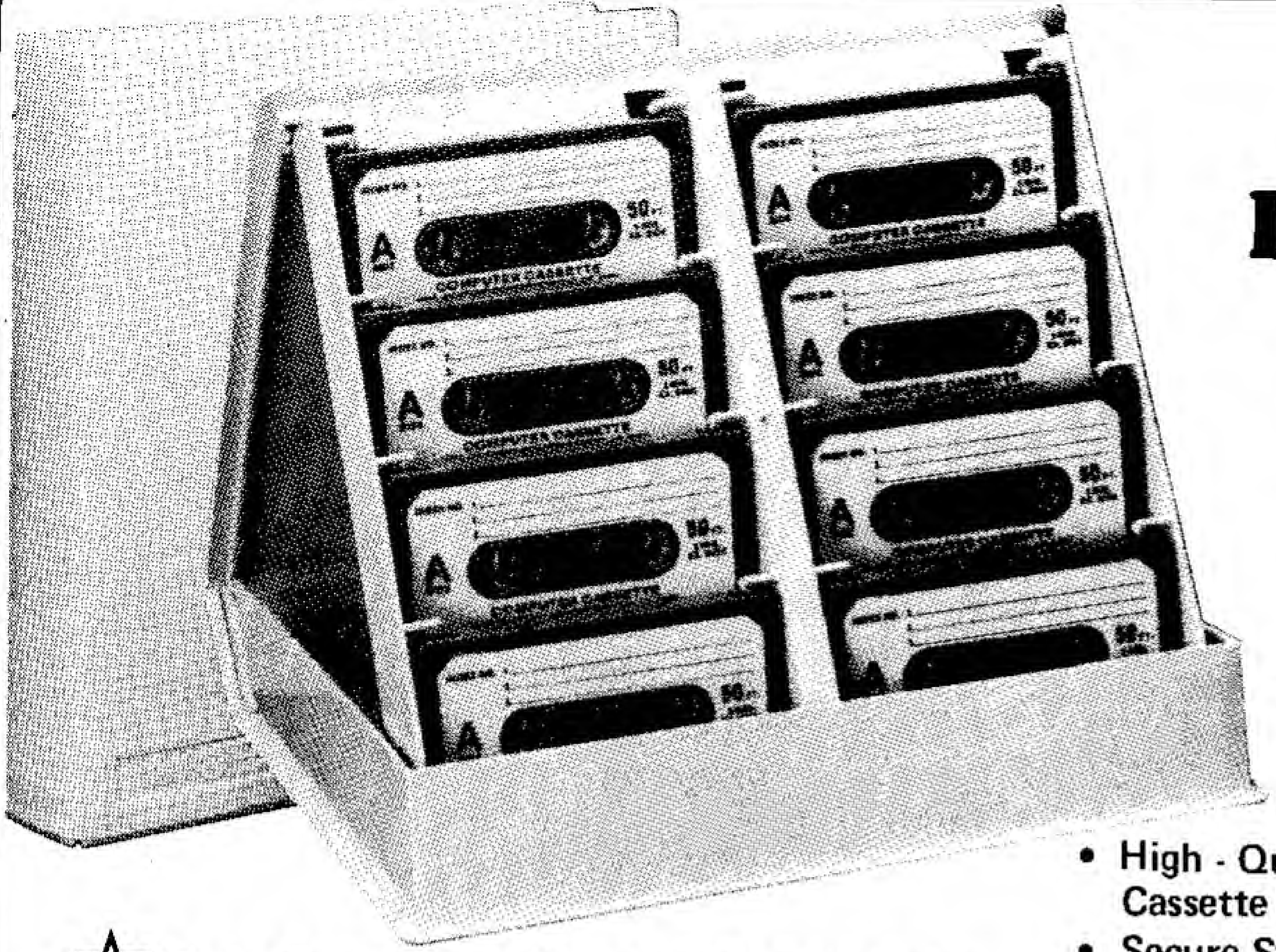
**Special Price: ONLY $49.95**

Add $2.00 for Shipping & Handling. Use the bind-in card in the back of the Magazine for your convenience in ordering. Telephone orders accepted if charged to credit cards.

99'er-ware™
INNOVATIVE PRODUCTS FOR TMS9900-BASED PERSONAL COMPUTING

P.O. Box 5537
Eugene, Oregon 97405
Tel. (503) 485-8796

99'er-ware ALL PRODUCTS ON THESE PAGES MAY BE ORDERED USING THE BIND-IN CARDS NEAR THE FRONT & REAR OF THIS MAGAZINE



## DIGITAL COMPUTER CASSETTES with LIBRARY ALBUM

- High - Quality BASF Tape in a 5-Screw Cassette Housing for Data Integrity
- Secure Storage & Quick Convenient Retrieval with Protective File Album
- 8 Special 50-Foot Cassettes — Ideal for Building Up Your 99'er Program Library
- Economical As It Is Functional

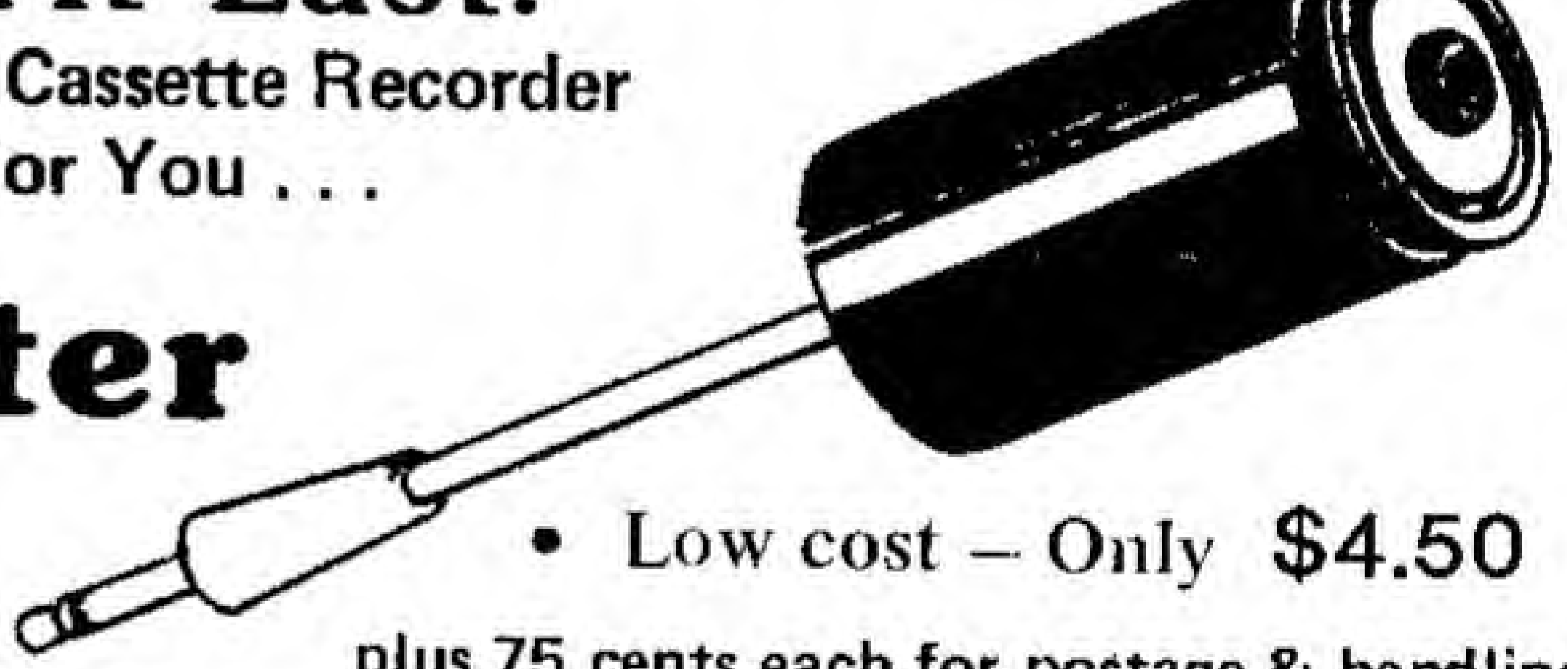
☆ Only $11.95 each; or 3 for $30.00

Add $2.00 shipping and handling for the first album; 75¢ for each additional album. Use the bind-in card in the back of the magazine for your convenience in ordering. Telephone orders accepted if charged to credit cards.

## Cassette Compatibility At Last!

**If The TI-99/4 or TI-99/4A Will Not Control Your Cassette Recorder Through Its Remote Jack, We Have The Solution For You . . .**

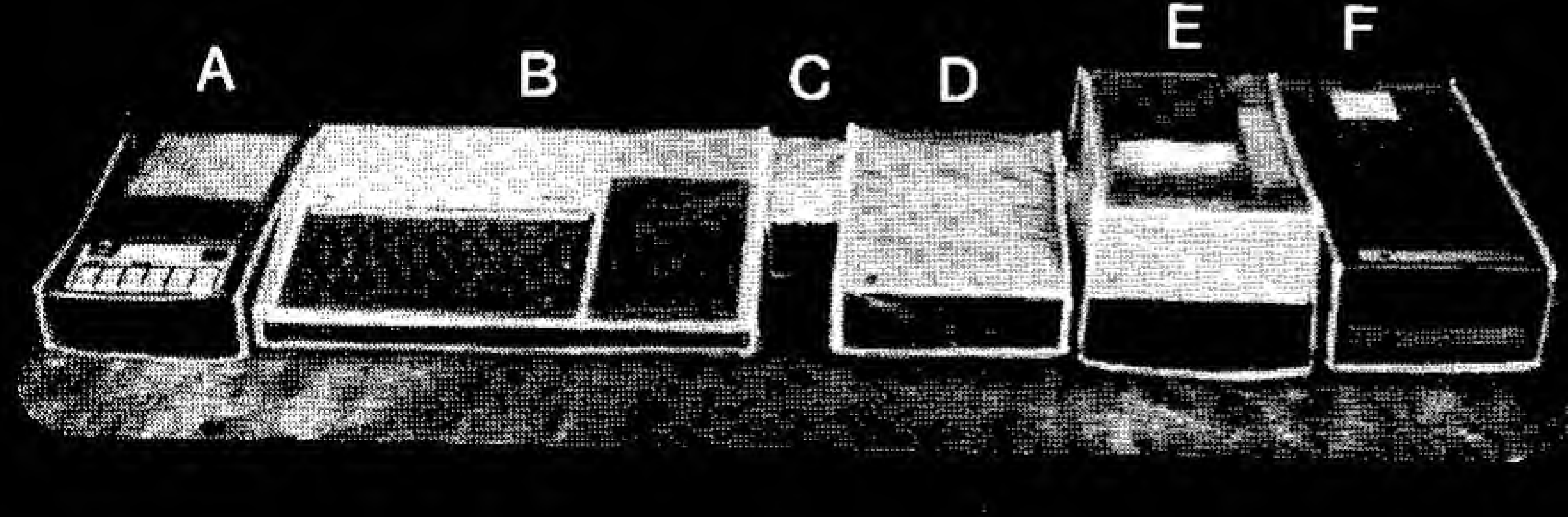
## The TI-SETTE™ Adapter



- Quickly connects between the black lead on the TI Dual Cassette Cable and your recorder's remote jack to establish compatible polarity.
- Low cost – Only $4.50

plus 75 cents each for postage & handling.

## DUST COVERS



Features:

- Equipment Protection
- Custom-Fit
- Handsome Appearance
- Antistatic Treated
- Quality Construction

| | | |
|---|---|---|
| A | Cassette Recorder Cover (1 size fits up to 10"x 6") | $4.95 |
| B | TI-99/4(A) Console Cover | $8.95 |
| C | Speech Synthesizer Cover | $3.95 |
| D | Peripheral Box Cover (Specify: 32K Memory Expansion, RS232 Interface, or Disk Controller) | $5.95 |
| E | Thermal Printer Cover | $8.95 |
| F | Disk Memory Drive Cover | $5.95 |

—Covers Not Shown in Photo:—

| | |
|---|---|
| 10" Color Monitor Cover | $10.95 |
| 13" Color Monitor Cover | $12.95 |
| Epson MX-80 Cover | $9.95 |

DEALER INQUIRIES INVITED

Add $2.00 shipping/handling for the first dustcover; 50 cents for each additional cover.

99'er-ware™ INNOVATIVE PRODUCTS FOR TMS9900-BASED PERSONAL COMPUTING

P.O. Box 5537
Eugene, Oregon 97405
Tel. (503) 485-8796

# TI Logo: IT OPENED A DOOR THEY THOUGHT WAS LOCKED.

*It opened a door to their minds.* The key: a Texas Instruments Learning Computer and TI LOGO, a programming language developed by TI and MIT.

In his inner-city, New York junior high classroom, teacher Steve Siegelbaum explains why it works so well. "When they use it, they think they're teaching the machine. In reality, it's teaching them how to learn. It definitely improves their attitude toward their other courses. Written and verbal expression improve–they're eager to show you, to tell you, what they've done."

Another teacher, Pete Rentof, adds, "What it fights is fear of failure–a mistake becomes a starting point. The whole learning process turns into a positive experience. It works."

The TI Learning Computer, with TI LOGO and many other educational programs, is equipped to help open doors in any classroom. Including yours. For information on this remarkable system, contact: Texas Instruments Customer Relations, P.O. Box 53 Lubbock, Texas 79408.

TEXAS INSTRUMENTS
INCORPORATED